SINCLAIR QL WORLD

The Premier Magazine for Sinclair QL Users

EVERY MONTH
£1.75
OCTOBER 1990

INSIDE FRONT COVER
4 6 7 8
9 11 12
15 16 18
20 26 31
38 46

CONQUEROR

Questions and Answers

DIY TOOLKIT

Toolkit On Disk

One Man's System

FROM WP TO THEATRE LIGHTING

Inside the QL

REAL NUMBERS INTEGERS AND

Software file

SPELLBOUND 2

ARCHIVE POWER

Wrapping up

Notebook

10
9 770951 933009

SINCLAIR QL WORLD

Editor
Helen Armstrong

Production Controller
Jayne Penfold

Designer
Jeff Gurney

Advertising Sales
Jason Newman

Magazine Services
Sheila Baker

Advertising Production
Michelle Evans

Group Advertising Manager
Richard Vaughan

Group Editor
John Taylor

Publishing Director
Ray Lewis

Managing Director
Peter Welham

Sinclair QL World
Panini House
116-120 Goswell Road
London EC1V 7QD
Telephone 071-490 7161
ISSN 026806X

Unfortunately, we are no longer able to answer enquiries made by telephone. If you have any comments or difficulties, please write to The Editor, Open Channel, Trouble Shooter, or Psion Solutions. We will do our best to deal with your problem in the magazine, though we cannot guarantee individual replies. Back issues are available from the publisher price £2 U.K., £2.75 Europe. Overseas rates on request. Published by Maxwell Specialist Magazines, A Division of MCPC Ltd. Sinclair QL World is distributed by SM Distribution, Tel: 081-677 8111. Subscription information from: MSM Subscription Dept, Lazahold Ltd., PO Box 10, Roper St, Pallion Ind. Est., Sunderland SR4 4SN. £21.00 U.K., £24.70 Europe, Middle East £25.80, Far East £27.60, Rest of World £26.20, U.S.A. $45.00. Airmail rates available on request. Typesetting by Adtec Typographics, Britannia Court, Basildon, Essex. Tel: (0268) 591110. Printing by Cradley Print. Sinclair QL World is published on the third Thursday preceding cover date.

© COPYRIGHT
SINCLAIR QL WORLD — 1990.

CONTENTS

OCTOBER 1990

9 QL SCENE ● QL Conference on bulletin board

10 OPEN CHANNEL ● A sort report

12 TROUBLESHOOTER ● Designing for speed

18 CONQUEROR QUESTIONS AND ANSWERS ● Emulator tips

20 ARCHIVE POWER ● The fourth and final part

25 BOOK PAGE ● Two books for programmers

26 DIY TOOLKIT ● Now Toolkit routines on disk

31 ARCHIVE DATES ● Rearrange dates with Archive

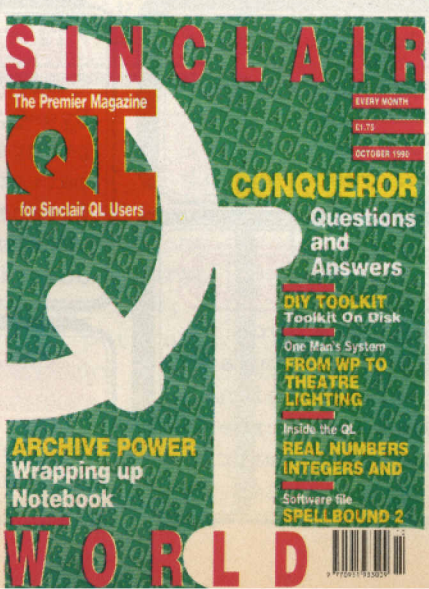
34 ONE MAN'S SYSTEM ● Wordprocessing and theatre lights

38 INTEGERS AND REAL NUMBERS ● As the QL sees them

42 SUBSCRIPTION INFORMATION

44 SOFTWARE FILE ● Spellbound 2

48 MICRODRIVE EXCHANGE



NEXT MONTH

JUST FOR STARTERS
A simple approach to using Abacus

PROGRAMMING IN C
Continuing the series started in the September issue.

QL SCENE

QL Conference on BBS

QL World has been given a list of bulletin boards which carry the International QL Conference. it is echoed between these bbss so that a message written on one system will appear on the others within a couple of days. Local phone numbers aree given.

Fourth Dimension BBs (UK) 0202 600305, Fidonet node 255/26; TF services (UK) 071 702 2379; Syncnet BBS (Holland) 035 237 178; Fidonet node 283/500; Ku-El Tel (Holland) 01650 37105; Fidonet node 285/102; Jamten-TCL (Sweden) 0642 10300, Fidonet node 202/602; PIX (Sweden) 031 960447; Fidonet node 202/300; EDKX 1 (Sweden) 08 719 5789, Fidonet node 201/108. All systems use 8 bits, no parity, and take 2400 and 1200 baud. The Swedish bbss operate in Swedish.

For more information contact **Michael Cronsten, Sysop, Jamten-TCL, Soere 1073, 83030, Sweden.**

Italian Meeting

The Second Italian QL Users' Meeting is taking place in Italy on 27th and 28th October.

The venue is expected to be the Centro Congressi Di Maderno, 5 or 10km from the Villa Alba, Lake Garda, last year's northern Italian site.

Attractions are expected to include talks and demonstrations from users and suppliers, a bring and buy stand, and advice about accommodation, booking etc, for foreign visitors.

For information please contact **Eros Forenzi, Via Valeriana 44, 23010 Berbenno (SO), Italy, local telephone 0342 492323, or Giovanni Zane, Viale M E Bossi 39, 25087 Salo' (BS), Italy, local telephone 0365 40102, local fax 0365 520184.**

New Family

Programmer Chris Boutal, already the author of *Archive* routines for genealogical researchers, has produced a compiled SuperBasic program for QL users interested in recording their family trees.

Called *QL_Genealogist*, the program is available on 3.5in disk, with a substantial and clear A4 manual which explains how to build up a family tree, and how to use the program. The disk as supplied contains the program as an obj file, along with a boot file (which needs extra memory to run), and an 128K version of the program for use on an unexpanded QL. There are demonstration files available containing the 30 generations of British kings and queens.

Users wanting to obtain the 128K versions on microdrive are asked to send a formatted cartridge when ordering — the disk is supplied as well as part of the standard package.

QL_Genealogist costs £19.50 from **Chris Boutal at 42 Charwood Road, Wokingham, Berkshire RG11 1RY.**

Ero

A short correction to the program *Ciphers*, from *QL World* August 1990: line 240 should end with:

allowed)": b2

and line 460 with:

to Quit"

Eras

A small amendment to last month's short Prog, *Dir_to_Archive_Bas.* The init routine at line 710 now reads:

719 WINDOW 512,256,0,0: MODE 4: CLS

Erat

And now a correction to *Programming in C*, page 23, in last month's issue. The backslashes were omitted from sections of text and one line in Figure 2. The paragraphs should read:

"Note that the first call to printf includes the characters \n in the string that is displayed. Characters preceeded by a \ are known as escape sequences, and have a special meaning. Some of them are:

Issue 2 August 1990

FT.ComClub

Newsletter of FT.ComClub, Fleet Tactical Command's User Group



\n — print a newline character

\t — print a TAB character

\f — print a FORMFEED character

\\— print a \ character"

\b — print a backspace character"

The missing slash from the last line of the text block in Figure 2 should read:

you use \n at the end

Our setters are looking into ways of preventing the \ escaping in future.

Eurofair

The European Microfair '90 takes place on 6th October in the sporting hall of the EuroVolleyCentre, Vilvoorde, Brussels, Belgium, as announced in last month's *QL World.* For information contact **Jacques Tasset, Aarlenstraat 104, B-1040 Brussels. Local phone 02 233 12 22.**

FTC Updates Sighted

Issue 2 of *FT. ComClub,* the *Fleet Tactical Command* User Group's small newsletter, has just appeared, carrying forward news of Fleet Tactical Command V2, which is now under development. V2 is scheduled for availability in "a couple of months", but the authors also say that they cannot develop the program as fast as they would like owing to the difficulty of covering costs on such a complex program.

Some of the projected developments are simplified TEXT command entries, and a TOW (ship) option which will prolong ship life. The "bad news" — their words, not ours — includes weather situations up to Gale Force 10, crew fatalities if hit, and greater damage sustained if attacked in a low state of readiness.

Some scenario changes have been made, and there is considerably greater flexibility all round regarding transfer of crew, replenishing stores and fuel, emergency cease-fires and infringement of neutral zones, to name but a few Small wonder that professional users are looking closely at Fleet Tactical Command.

OPEN CHANNEL

Open Channel is where you have the opportunity to voice your opinions in *Sinclair QL World*. Whether you want to ask for help with a technical problem, provide somebody with the answer, or just sound off about something which bothers you, write to: Open Channel, Sinclair QL World, 116/120 Goswell Road, London EC1V 7QD.

Sort

Neil Taylor's article on sorting was avidly read here, as sorting is a must in my Basic programming. I am recording churchyard lichens, and use three arrays, one for Latin names [lat$(x,36)], one to hold four code numbers for church, substrate, etc. [lik%(x,3)] and an index array [ix%9x)].

After a batch of names is entered, in order to save cartridge space, the entire list, now exceeding 4,000, is sorted by two parameters: Latin name and church number combined. Then all duplicate names and even duplicated generic names are reduced to zero and the array saved. When re-loading, the missing names and genera are replaced, and because the index is also sorted, the array is filled in the original unsorted manner, church by church, 4000 names takes about 45 minutes to sort.

I compared Neil's sort with a bubble sort, and also with Marcus Jeffreys' recursive quicksort which appeared in the August 1985 issue of *QL World*. The results were:

Bubble 171 seconds
Neil Taylor 106 seconds
Marcus Jeffrey 36 seconds

All of these would have been approaching 30% quicker but for the fact that I print one of the incrementing array lines to screen to watch progress.

No-one, so far as I am aware, has yet bettered Marcus Jeffrey's sort. The need to have an extra array line with an entry such as 'Zzzzz zzz' or '99999' is no problem. The problem that I can see with Neil's sort is the need for a duplicate array in which to place presorted items. If one's ram is already three-quarters full, I don't see how there can be room for another similar sized array.

I enclose the test program. The command 'setup' fills the initial array, 'slow', 'start' and 'fast' initiate the sorts. 'dup' refills the initial unsorted array before trying another sort and 'see' enables the results to the screen.

Don Smith
Westfields
Kirkymoorside

Editors comment: Send an SAE to QL World *if you would like a copy of Don Smith's listing.*

Neil Daglish will copy his program for anyone who sends an mdv or 3.5in disk plus a self addressed envelope, to him at the Education Department, Victoria University, PO Box 6000, Wellington, New Zealand.

Keyboard

The January issue of *QL World* prompted me to write on two points. Firstly, in Daniel Blaum's article on the Qualsoft terminal emulator, he refers to receiving at 4800 baud as "the QL seems to have difficulty receiving at 9600 baud". The reason is that he has specified one stop bit for STarTERM and Procomm (see page 14 of the Concepts section of the User Guide: "If the QL is set up to 9600 baud . . . at least 1½ stop bits are required.").

Secondly, how has John Acielo interfaced his XT-type keyboard to his cased-up QL? For that matter, have any readers obtained information on connecting PC-type keyboards to QLs by means other than the Schon or ABC interfaces?

On a slightly different topic, have any readers had any success networking QLs and Spectrums with Interface 1?

Mike Fleming
Tamworth
Staffs

Editor's comment: Interfacing PC-type keyboards is a popular subject now that the Schon and ABC interfaces are no longer freely available. QL World *would like to hear from readers who have tackled this type of project successfully.*

Typewriter

Quill is a very good word processor, but it is a complicated procedure loading Quill when all that is wanted is to type a few lines for, say, a short memo or to address an envelope. Your readers might be interested in the following short program which makes the QL plus printer operate like a typewriter. It was written for the Epson LQ400 dot matrix printer with a parallel interface, but it should work with any printer, although it may be necessary to change line 120, which sets left and right hand margins, for a different printer.

```
100 OPEN#3, SER1
120 PRINT#3, CHR$(27);
  "1";CHR$(6);CHR$(27);
  "Q";CHR$(75)
130 MODE 4
140 WINDOW #2,482,246,15,
  9,BORDER #2.1.1.7
150 WINDOW 416,235,45,19,
  PAPER 2:INK 7:CLS
160 REPeat getline
170 INPUT a$
180 PRINT#3, a$
190 END REPeat getline
```

It is not possible to make the printer type letter by letter, but

Editor's notebook

The news that Simon Goodwin's past DIY Toolkit programs are being made available on disk will be pleasing to the serious programmers. Uneasiness about the magazine's future prior to the Focus troubles and the difficulty of organising the programs prevented Simon from making them available earlier, but he has now done so. We hope also to hear very shortly about the fate of the Microdrive Exchange and whether we can restore it to its earlier status. Either way, the time will soon come to start developing it again.

On the same day I receive news from Bryan Davies that PDQL has sent him a long-promised copy of Hardback, and a letter from the Birmingham City Council Consumer Protection Division, asking us if we would forward to them information about any complaints we receive about the company. They can be contacted at 155-157 Corporation Street, Birmingham. QL World still wants to know about any problems and solutions, of course.

Lastly, thank you to the Family tree users who have sent suggestions for getting screen dumps.

this program types line by line. Each line is displayed on the screen and can be edited to remove errors. Pressing ENTER causes the line to be printed and moves the cursor to the start of the next line. The WINDOWs are set to give a half-inch margin at each side of an A4 sheet of paper but these and the printer margins could be changed to suit smaller sheets.

T Ashcroft
Jesmond
Newcastle upon Tyne

C-Port

The latest problem I have had with PDQL concerns an alleged product called *SuperBasic C-Port*. For months now the mild-voiced man to whom one speaks on the rare occasions that one can get past the Ansaphone has been assuring me that the product is all but ready to ship (but if it wasn't ready in December, why did he advertise it?) and indeed on the last two occasions he assured me that it was now ready to ship and only needed copying onto a disk and putting in a Jiffy bag.

What he has done by misleading me for six months is to divert me from seeking an alternative solution to my problem of how to get a particular piece of software over to the IBM environment. As a result, I would caution anyone seeking to enter into any business relationship with PDQL. For support one needs reliability, and PDQL is, in my experience, totally lacking in that commodity.

Might I make a final offer through your columns. If there really is a piece of software out there called SuperBasic C-Port, and if there is anyone who has a legitimate copy (one of the Beta testers if they really do exist), or better still, the authors read this, would they please contact me at the above address. I will pay costs to the first one to send me a copy. I will also guarantee that, should I receive a copy in this way, I will make my offer to purchase a legitimate copy at the full price from PDQL irrevocable, should that outfit ever get around to sending me one. I cannot say fairer than that. I am not trying to promote 'pirate' software.

The real trouble with the QL has been that a brilliant machine has never taken off because it is, in part, marketed and supported by wallies like PDQL.

J A Davis
54 Woodlands Road
Bookham
Surrey KT23 4HH

Editor's comment: Private users and especially businesses suffer greatly from misleading claims by suppliers, not just in the QL community, with the greatest difficulty in establishing any rights of redress.

The situation between PDQL and software authors on their lists is not known to us in any detail but we have heard from at least one author who would like to remove his products, and others who have expressed great concern. Copying non-pd software for other than backup purposes is always a breach of copyright, but the exclusive copyright does not always rest with the publisher. If anyone with executive power over this program is reading this, QL World *as well as J A Davies would like to hear from them.*

Listings

On the subject of user-friendly listings: many magazine listings seem to be error-ridden. To debug them takes some time as often the listings are optimised down to minimum length. The only simple solution, which I sometimes adopt, is to write programs in long-winded transparent style. When these are trouble-free I then optimise them and store both listings. If magazines were to print both "near-English" and "abbreviated" versions, this would greatly facilitate comprehension, development and exchange of routines.

With the QL, "near-English" programming is the form advised in the User Guide. Luckily, English listings rarely use the interactive "i" variable (as is so often the case on the Continent) but one too often meets "L", "I", and "O".

Regarding formal parameters, these are a bug-bear, a fact seldom referred to in QL literature. The only way to use them efficiently is to pass the formal value to a local vairable at the beginning of the routine, and to return values using a global variable such as OUT. It is worth noting that while formal parameters are type-independent on e ıtry, it is impossible to return alphanumeric strings unless the routine name is suffixed with "$". There is scant mention of this in any documentation!

Regarding division by zero: someone, somewhere, decided that this was "illegal". In graphics routines, especially using 40, division by zero is often necessary, and can be replaced with no effort by dividing by minus one over infinity.

Finally, why bother to buy a QL? In my own case, I bought a JM in 1984 in France to learn programming, and to replace a TI 57 which I used in land drainage surveying. At the time, various "advisers" said it was impossible to use the Psion 4 in accounting, graphics projects, etc. The company therefore bought other machines and tailor-made programs that neither give complete satisfaction nor can be understood by the "advisors". So my QL offers a few games for the kids and allows myself to develop greater knowledge of problem-evaluation.

As a leisure activity, programming beats crosswords hands down.

Stephen Poole
Aube
France

Mice

Mike Lloyd (Software File, August 1990) says that our QL is only half the beast without QPAC-2. He also notes tht "mice are rather rare in the *QL World*". Both true: but if a trackerball is used in place of a mouse, the improvement is even more spectactular. The Atari trackerball is unnecessarily large, but it fits instantly in place of the QIMI mouse, and works wonders with bar-menu programs like *Page Designer 2*, for example. Mine cost £15, secondhand.

C R Oswin
Christchurch

PS Windows/icons/mouse/programs = WIMP. Bar-menu/trackerball/programs = BARMPOT

Cover

You ask for comments on the new cover. It is a little difficult to judge from the mono preview, but I'm afraid my initial reaction is tht I'm not over-impressed. On the other hand, I am not really worried about the cover, particularly since I get the magazine on subscription and therefore do not need to have my eye caught by it on a newsagent's display.

I am, however, interested in the contents and, irrespective of my queries to the *Psion Solutions*, I think they are improving. There are fewer errors in listings these days. There are still a few errors in typesetting, notably a letter in the August issue decrying the use of the letter l as a variable, being confused with the number 1. I do agree with the writer, though, and, while recognising the problems, would be inclined to return such listings to the authors for editing, as I also would those listings which purport to be from knowledgeable people who, nevertheless, can't be bothered to set up translates property but set the dip switches to get the £ sign and end up with listings giving channel £1, etc.

Ken Davies
Silverdale
Lancs.

Editor's comment: We wanted to run the new design in colour, but we ran out of colour sites and had to swap the abstract base artwork for the paler current month's cover. We considered the point of view that the cover doesn't matter to subscribers, but we find it improves the handling qualities and usefulness of the magazine to have a distinct cover each month, even if it is mainly a variation in colour. Also, of course, newsagent buyers are an important part of our readership. Not everyone wants to buy every issue, but they want a chance to look at it.

That letter must have slipped through at the last minute, with the result that somebody will be having a good laugh at my expense. I am in agreement that it causes fewer problems if your 'default' settings give accurate listings rather than accurate pound signs.



TROUBLE

APROBL

Bryan Davies considers the hardware and software design options which could make the QL a faster machine.

There has been some discussion recently about possible upgrades to the QL. As the subject is in the air once again, what about sending in your comments regarding what you would like (reasonably) to see in either add-on boards or a new machine? One obvious approach is to make a go-faster board that could be fitted to existing QLs, but this approach does place quite severe constraints on designers and they would probably prefer to be able to design a complete new machine. Features which would almost certainly be present on a new machine are 40 MB (or more) hard disk, 1 MB (or more) ram, 68020, 68030 or 68040 cpu chip, and an improved operating system. Don't neglect to say how much you would be prepared to spend!

Speed

The QL is often complimented for its good design, but speed is not one of its best points. Not that it is particularly slow, but it certainly looks sluggish when compared to some more modern systems. Speed is very much a perceived thing. As with 0-60 mph times for cars, many benchmark test results are just plain useless to the ordinary user as a serious guide to how the computer will perform. Since most of our readers seem still to be using *Quill* and microdrives, the clock rate of the central processor is not the factor that matters most to them. The slow routines used in Quill, for putting text on the screen and for accessing the drives, are of much greater importance.

Comparing Quill on a PC/AT with Quill on the QL, using hard disk on the PC and floppy disk on the QL, the QL looks terribly sluggish, even with *Lightning* and *Turbo-Plus* giving Quill a boost. What matters to someone who wants to create sizeable documents is the speed with which the cursor can be moved around, and the loading and saving times. It's not really comparing apples with apples, though. Quill was rewritten for the PC, with the experience of the QL version available, and runs much faster because of that.

Almost any hard disk operates faster than any floppy disk. My PC has a disk-caching routine which raises performance well above standard; the cpu itself has a nominal clock rate of 10 MHz, which is not that much greater than the 7.5 MHz QL. Any design for a successor to the QL has to utilise much faster screen– and disk–drivers, and the poeple who write programs for it need to be smart in using the improved drivers. There are programs which make the QL look speedy. *Flash-Back* is an obvious example. *QTyp* makes spelling checking as fast as you can respond to the prompts, as does V2 of *SpellBound*.

Here are some suggestions from a hardware designer who seems to know what he is talking about:

1) The best way of making a go-faster QL is to design a (software) QL emulator to fit into an existing, fast computer, such as a Macintosh with a 68030 cpu. The trouble with this is that few QL owners would contemplate spending several thousand pounds on a Mac to speed up QL programs (the suggestion was, being able "to run Quill four times as fast").
2) Produce a 68030 computer from scratch, with a rewritten QDOS. Much of the QL's activity relies on at least one other chip, apart from the cpu; the 8049 would have to be replaced, and its replacement might have to handle a larger share of the load.
3) Design a hardware QL emulator, to fit on a PC card. This would also require a fair bit of new software. PCs are a lot cheaper than Macs, but they're not going to be fast at emulating the QL if they cost under a thousand pounds.

There is already a good QL emulator running, on the Atari ST, and no doubt some readers have got this desirable combination. The fact that it doesn't appear to have sold in large quantities suggests the price (of ST plus emulator) is too high to tempt many QL users. The Thor XVI was basically a good try that went wrong for largely non-hardware reasons. Had this machine been marketed well, at a much lower price, it would have had a fair number of takers amongst existing QL owners. The impression given at the time was that CST was not interested in selling it to QL owners, and the price put it up against good PCs.

Unless someone comes up with more than bright ideas for new hardware, the only way to get better performance seems to lie in following the route already well-trodden on other computers — keep churning out the "accelerator" packages, both software and hardware. The *Minerva* rom has proved popular, and it gives speed improvements in certain areas, as does Lightning (on disk or rom).

If the programming experts now feel the screen is being handled as fast as is reasonable, what about looking at faster access to drives and memory? Is disk caching feasible, and worthwhile (bearing in mind the existing slave block activity)? Can the existing memory be cached (maybe only any add-on ram)? There seems to be a resurgence of interest in making the QL go faster (maybe memories of the Futura project have faded), and an add-on board using a 68020 cpu has been demonstrated recently, showing a commendable turn of speed. That idea should be a relatively safe route, but others favour going straight to the fastest processor currently available in the 6800 series, the 68040. Whether or not the necessary hardware and software expertise to utilise these chips effectively is available in the QL world remains to be seen, but it does look likely that some hardware will be available during the next year or so to run existing programs at a much faster rate.

International

As a QL user, you are part of a very wide-spread community. Letters received in recent months include ones from New Zealand, the USA, Germany, Belize, Italy, County Down, as well as from Taunton in SW England to Edinburgh in Scotland, with a good sprinkling from places in-between. There are many clubs, of all types, around the world and the membership of many of them may be numbered only in hundreds. Our 10,000-plus active fraternity is healthy enough to keep going for a long while yet. You may have noted that the QL Users' Group Quanta is now into its seventh year and membership is around 2,000 — as high as it has ever been. It is still possible to talk about the QL to some of the people who have worked with it from early days, and are still involved with development of both hard-

SHOOTER

EMSOLVED

ware and software for it.

No doubt many users, and some suppliers too, consider there is no new software of note needed for the QL, but that is debatable. There is still no integrated suite of utility programs available to handle all housekeeping tasks, on hard disk as well as cartridge and floppy. The lack of suitable routines for handling hard disk files can make setting-up a hard disk quite a chore. *QPac* is, perhaps, the best piece of QL software for handling such a task, but it is not so much a suite of programs as a concept that you buy with QPac, and that may not be what many users want.

Stand-alone

The kind of thing I am thinking of is a stand-alone package, that could be used with any existing system set-up, one that would allow the user to go simply, and quickly, to any hard disk sub-directory, see all the files listed, and perform normal operations, such as Copy and Delete. It would be essential to be able to display a "tree" view of the directory structure. The present drive software for the Miracle hard disk makes it (for me) very confusing to find out what sub-directory you are in, and to navigate to another one. More commands are needed, such as Rename for sub-directories as well as files.

While MS-DOS is far from being the be-all and end-all of operating systems, it would seem sensible to make present-day modifications to the QL operating system compatible with existing (and useful) MS-DOS functions. One program which might be expected to be very useful with hard disk is *HardBack & Finder*, by Chas Dillon, but it currently seems to be unavailable because of the troubles at PDQL. Hopefully, it will reappear before too long. A program which is available, and can tackle some aspects of hard disk housekeeping, is *Files 2*. With some modification, this program could do a competent job of handling most hard disk requirements; the writer is considering the project.

Going back to the subject of a disk-caching routine, it may be that QDOS already performs part of this activity anyway, but the use of slave blocks, but it seems possible that hard (and maybe floppy) disk access could be speeded-up appreciably overall. The effect of a disk cache on a PC is tremendous, making a just-acceptable performer into a distinctly-fast one.

The hard disk makes it difficult to live with weaknesses in software and hardware that one accepted previously. For example, we have had 896 KB of memory available for a year or two now, but no sign of more — leastways, not in a form that is trustworthy. Some users will not be too worried about lack of ram memory, but a significant number must want more (if only to get away from frequent resets). The program-switching software currently available for the QL works only within the ram area, but some users now have 40 MB of hard disk, onto which programs not currently being used could be switched. This concept has been used on the PC for about five years now and can transorm a basic 8086 machine into a genuine multi-tasking one. (To those who think this is not possible, I suggest reading-up on the subject of expanded memory.) Task-switching (having only one task running, with others "frozen" in the background) is fairly straightforward, with basic hardware, but multi-tasking does require expanded memory. The QL always was in advance of the PC in respect of being able to handle several tasks concurrently. All it needs now is for that ability to be put on one side (do you use it?) and an effective task-switching program to be written that uses hard disk to store any applications programs which are currently not being used.

Tactical link

CGH Services advise that it is the *QL Technical Review* which has got to Issue 3, not the QL Leisure Review, which is yet to be published (as of early July). Further modifications are being made to *Fleet Tactical Command*, including making it possible to link two computers via modems, and to run the program on (IBM-compatible) PCs. Some minor bugs have been removed. The "alpha test" versions of an important new program from a well-known supplier look encouraging; the presence of several bugs is no surprise, given the scope and capability of the program, and reported bugs are being fixed quickly. Having mentioned bugs, it might interest users who are unfamiliar with other computers to know that programs which sell in the millions of copies (and cost several hundred pounds each) can have dozens of bugs in them, even after several upgrades. The PC WP program with which I am most familiar certainly has dozens, but none of them cause me great problems. To some extent, the complexity and size of current programs make the presence of some bugs virtually inevitable.

Making comments about printing from Quill seems to inspire readers to send in suggestions on the subject, which is the way we like it to be. If you didn't like the way of sending and ESC code to the printer mentioned in the August issue, here is another one to try, courtesy of Stephen Meech. Hold down the CTRL key and tap the ; key — this produces the CHR$(155) code. How your printer sees this may vary, but it is likely that an "Epson-compatible" printer will interpret this code as an ESC, after stripping the MSB (most-significant byte) from it. If you follow the 155 by the appropriate code(s) for the function you want, the printer should switch the function on. This will work from SuperBasic, or from within your word-processor (provided the CTRL; code is accepted). The *QL User Guide* shows decimal 155 as producing, on the screen, a u with a hat (circumflex accent) on top of it. If you follow that by -1 (dash 1) you will have the codes for turning underlining on. Likewise, CTRL; followed by M will give the code to switch Elite (12-pitch) characters on. I've not tried this in other programs yet, but it certainly works from *The Editor*. One negative aspect is that you have to put more codes onto the screen than you would if you used Translate functions, but the problem with Quill is that you may run short of Translates, so this is a way of getting extra functions out of the printer when that situation occurs. You also have to remember that, when non-printing codes are removed at print time, the following text moves a corresponding number of spaces to the left, which may upset spacing and justification. This technique for printer control is applicable when printing from Abacus also.

Readers' Letters

J. Roy Goodall in Belize asks for a way of transferring Archive _dbf files to-and-from the Psion Organiser. Has anyone got a way? **J K Easlea** asks if it is possible to alter *QL Home Finance by* **Buzzz** to make use of more than the basic 128 KB memory. **H F Banks** would like to know the present whereabouts of **RSD Components**, previously of Ware in Hertfordshire. **Brian Hedge** has offered to provide **J S Hay** with a working copy of the Rename routine from the January 1987 *QL World*.

Sector Software reports that a complaint from **E Stocker** concerning damaged goods has been dealt with.

What more can be said about **PDQL**? Some of the money owing to Miracle Systems has been paid to date; by the time this issue of *QL World* appears, the total debt should have been cleared, but apparently only because Court action was taken. I have received just one letter of complaint about PDQL in the past couple of weeks. No "letters of satisfaction" arrived from existing complainants. **James Costello** of New Jersey in the USA has not received PDQ-C six months after ordering it. He has spent almost as much as the program cost on making 'phone calls to PDQL, to no avail. A copy of *HardBack & Finder* promised to me by PDQL several weeks ago has not materialised.

Assurance has been received from **TK Computerware** that *Turbo Quill* and *Turbo-Plus Quill* are still available, from them, and will continue to be so in the foreseeable future.

Keyboard Products report that there has been some delay in delivery of PS/2 keyboards, due to redesign of the internal circuit board, but work has been completed and deliveries should have been restarted by now.

It is unusual to receive a complaint about a non-UK supplier, but D. Stewart says he has lost £195 which he sent to ABC Elektronic for one of their Mega Ram units. The unit was not supplied "due to technical problems", and the proprietor of the company seems to have moved on, leaving no forwarding address. The products of ABC have been taken over by Jochen Merz, but he obviously cannot be expected to supply items for which the payment has gone elsewhere.

Leon Heller has written to advise that MetaComCo ceased business some while back (the staff went off to do other things, rather than the company going broke), but Leon has managed to get a replacement for the missing *MetaComCo C* cartridge and has been good enough to send it to Ivan Zorzin (see his plea in the July issue). Some of the MetaComCo staff joined companies working on projects using Transputers, which have been one of Leon's main interest for the past few years.

B.F. Boote asks for help with a disc fomatting problem. He has two 5¼ inch drives and a SuperQ interface, and one drive formats to only 720 sectors, whereas it (and the other drive) normally formatted to 1440 sectors. The drive has been checked and pronounced fit. Any ideas? David McKail has printing difficulties from his Thor, and is finding contacting Thor International difficult. He has Argos version 4.21 and XChange 3.90 — can any readers report experience of using these versions?

A word of caution concerning making claims on credit card companies for products which have not been received. Before coming to the conclusion that the supplier from whom you ordered the goods never shipped them, leave adequate time to allow for all reasonable eventualities in the delivery process, and then advise the supplier in writing that you want a refund. It will take some time for the refund authorisation to be processed by the credit card company; if you contact them first, or soon after you complain to the supplier, you may get the impression that the credit card company then credits your account as a "goodwill gesture", and that the supplier has not authorised the credit. As the credit card company will probably not say why credit has been given, you may never know that the supplier has "done the decent thing".

All suppliers are not,by definition, bad. It is unfortunate that SUB and PDQL have combined this year to give a poor impression of QL suppliers in general. Although there are some features of other suppliers which you may feel are not entirely praiseworthy, the well-known suppliers that remain on the scene have generally-good reputations, and have been around long enough for one to have reasonable confidence in them. Even when one is treated in some way that is felt to be unsatisfactory, it is desirable to bear in mind the almost daily reports in the newspapers of major companies leaving customers and suppliers in the lurch, with far greater losses than we are ever likely to sustain from orders for microcomputer goods.

QL SUPERTOOLKIT II by Tony Tebby

Over 118 Commands:— Full Screen Editor, Key Define Print Using, Last Line Recall, Altkey, Job Control, File Handling, Default Directories, Extended Network.
16k Eprom Cartridge Version ..... @£ 24.15d
Configurable Version on Microdrive ..... @£ 24.15d

MIRACLE SYSTEMS PRODUCTS

OK Disc Interface (inc Toolkit II) ..... @£ 99.82b
QL Centronics Printer Interface ..... @£ 28.75d
QL Expanderam 512K Thrucard ..... @£ 99.99e

QL HARDWARE

Single 3.5" Disc Drive & (Own PSU) ..... @£103.50a
Dual 3.5" Disc Drive & (Own PSU) ..... @£188.60a
3.5" DS/DD Discs – 10 off ..... @£ 9.20c
Q POWER REG. The only real solution to your QL overheating (switched mode power supply run cold) ..... @£ 24.15c
QL Keyboard Membrane ..... @£ 11.50d
QEP III Advanced Eprom Programmer ..... @£121.90d
Care Eprom Cartridges each ..... @£ 5.75c
ULA CHip ZX8301 ..... @£ 15.64c

PRINTERS

STAR LC10 Mono ..... @£179.40a
STAR LC10 Colour ..... @£224.25a
CITIZEN SWIFT 24-pin Colour ..... @£356.50a

SOFTWARE 87 (State MDV or Disc)

TEXT 87 V.3.00 ..... @£ 60.00d
FOUNTED 89 ..... @£ 15.00c
FOUNTEXT 88 ..... @£ 25.00c
TEXT 87/FOUNTED 89/FOUNTEXT 88 ..... @£ 94.99b
2488 PRINTER DRIVER ..... @£ 15.00c
Upgrade to Text 87 V.3.00. Return old copy together with ..... @£25.00d

MONITORS (Price including lead)

Philips BM7522 Amber Hi-Res ..... @£ 97.75a
Philips CM8833 Colour Med-Res ..... @£253.00a
Philips AV7300 TV/tuner for above ..... @£ 69.00b
Philips BM7502 Green Hi-Res £97.75a

HOW TO ORDER: **ALL PRICES INCLUDE VAT**

By Post. Enclose your Cheque/PO made payable to CARE Electronics. Or use ACCESS/VISA. Allow 7 days for delivery

TONY TEBBY SOFTWARE (QJUMP)

QLFP (Micro/P disk interface upgrade) ..... @£ 14.95d
QMON II ..... @£ 23.00d
QTYP Type/Spell Checker ..... @£ 29.09d
QPAC 1-Desk top accessories, calender, alarm, calculator, typewriter, digital clock Sysmon ..... @£ 21.85d

ZITASOFT SOFTWARE by Steve Jones

LOCKSMITHE copies M/DRIVE — M/DRIVE ..... @£ 14.95c
4MATTER + LOCKSMITH copies M/DRIVE — DISC .. @£ 23.00c
The above programs are not for use in the UK.

SIDEWINDER – High resolution printer driver prints full screens or parts of screens from postage stamp size to large banners. Prints sideways, invert, scale, mirror, text insertion.... @£ 19.95c
SIDEWINDER PLUS – (for expanded QL's) includes all the features of above, PLUS multiple label printing, desktop publishing files and printer driver for 24' pin plus LC10 and J× 80 colour printers. (Please state 3.5" disc or M/D).... @£ 23.00c
Upgrade to Sidewinder Plus ..... @£ 8.05c

HEAT TRANSFER RIBBONS & PENS

Print your own T-shirt Design. Just print on paper and iron onto your Tshirt.
Star LC10 NX1000, Rainbow ..... @£ 17.25c
Star LC10 NX1000, Black ..... @£ 11.50c
Epson FX80, MX80, LX80, FX100/ Okidata ML80 Citizen 120D/Star NX10, Black ..... @£ 11.50c
Small 5 colour pen set ..... @£ 12.65c
Jumbo 5 Colour Pen Set ..... £ 16.10c

PAINTER

The QL drawing program by PROGS. Fully multitasking ..... @£ 37.95c

READYMADE LEADS

RGB QL to Phono ..... @£ 5.75c
RGB 8-6 pin DIN ..... @£ 7.13c
RGB 8-7 pin DIN (Hitachi) ..... @£ 7.13c
RGB 8-7 pin DIN (Ferguson) ..... @£ 7.13c
RGB 8 pin to SCART (Euro) ..... @£ 11.50c
6-way PCC 25-way 'D' (Printer-Ser 1) ..... @£ 9.89c

QPAC II

QPAC II Main menu windows adjust automatically to size. Files, directory, view, print, delete, backup, jobs, pick, Rjob, sort, channels, things, exec, wake, buttons, Hotkey, Hotjobs. Fully multitasking, allows many programs to run at once
Requires min of 256k expanded memory ..... @£ 49.91b
Upgrade QRAM to QPAC II return master ..... @£ 29.90b

AS REVIEWED QL WORLD AUGUST ISSUE

CARE ELECTRONICS
OPEN
9am-5pm Mon-Thu
9am-4pm Fri

800 ST ALBANS ROAD, GARSTON, WATFORD, HERTS, WD2 6NL.
Tel: 0923-672102
Fax: 0923-662304

Please add carriage
a=£11.50 b=3.45
c=£1.38 d=£2.30

27 HATHERLEY COURT
HATHERLEY GROVE
LONDON W2 5RD

The Editor
Open Channel
Sinclair QL World
Panini House
116-120 Goswell Road
London EC1V 7DD

Doing away with microdrives

Dear Editor,

There has been a lot said recently, about what we should do when the supply of microdrive cassettes, already low, is finished. Personally, I gave up on microdrives a long time ago. I bought my QL when the price was about £400 and at least a quarter of the cassettes I purchased ran unsatisfactorily, for they could spin wildly out of control all night if I let them, without inputting anything at all into the QL itself.

So after consideration, I purchased Cumana twin 3.5 inch Disc Drives and 512K Expanderam extra memory. The interface for Cumana fits into the Expanderam body, which fits into the left-hand end of the QL, using altogether only one of the QL ports. From day one I have had after this installation (touch wood) evening after evening of trouble-free running. It was from a recommendation in QL WORLD some time ago that I made my choice.

In a recent issue of "QL WORLD" such a set up was described as unsatisfactory, because the Expanderam unit with the Cumana interface hung so far out from the QL that it was unsafe, but I purchased a board from the hardware store, which was on sale for use as a ready-made shelf. I had no intention of course, of using it as a shelf, but as a resting place for my QL, with its above-mentioned add-ons.

The board is 900 x 200 cm (or 35.5 x 8 ins.approx). With supporting lengths of 20mm. square (finished size) wood under the overlaps of the QL body, this is very convenient and the Disc Drives sit on it at right-angles, with the L. corner of my Taxan printer inside the angle.

Special Editor, Cue Print and Spellbound multi-task conveniently on one disc. Abacus, Archive and Easel are contained on another. My disc with Professional Publisher has room for no further programmes. Extra memory has now brought my QL up to 640K, the same as the IBM XT I use at my office but Professional Publisher will still only run comfortably with no more than 4 of the fonts resident at any one time. This, however, is really not a great setback, for exchanging fonts is quite a fast, simple operaton.

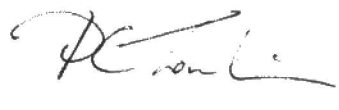
To me, my QL ugraded to this extent compares very favourably with the IBM XT at the office, but the non-extractable hard disc used there mainly because of security reasons, is a cause of dissatisfaction. Because I am maybe biased in any case for the QL, I think working with hard discs is not all it is made out to be, but I am not really a programmer.

Abacus is not quite such a powerful instrument as Supercalc on the IBM. But Special Editor on the QL is more useful and faster than Word Perfect on the IBM in every day service.

I would highly recommend an upgrade of this type, to do away with the worry of microdrives for ever.

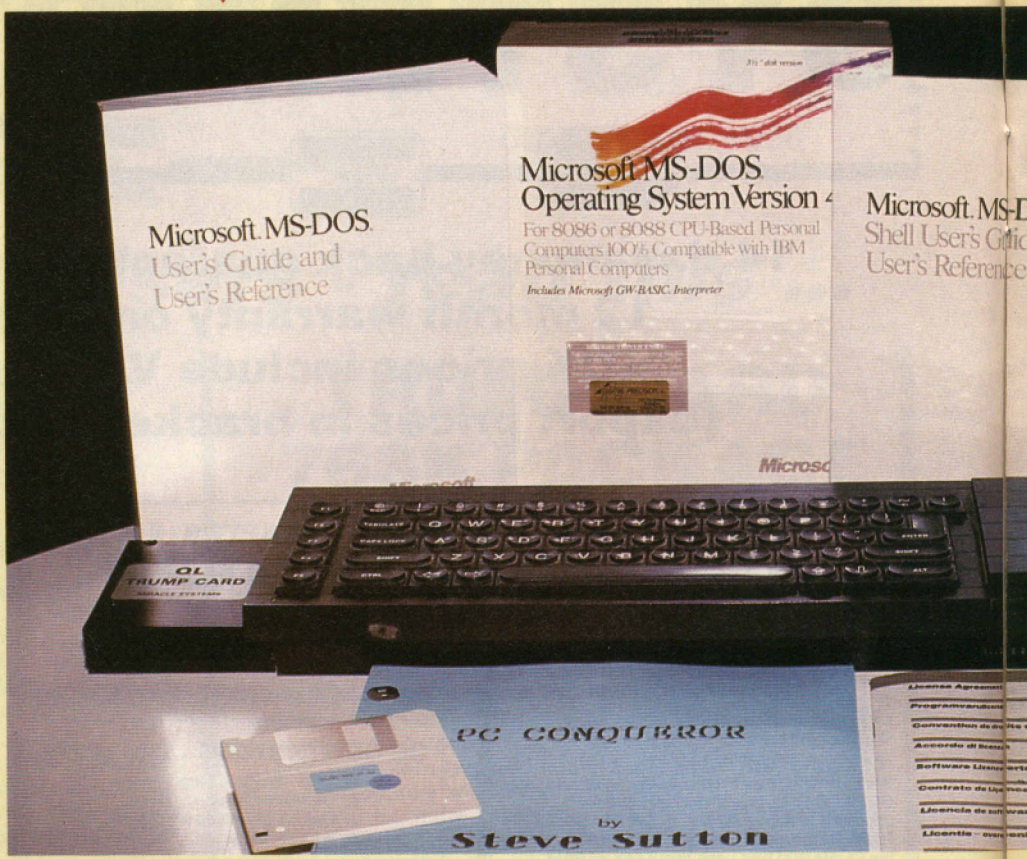
Kind regards,

P C Tomlin
Hatherley Grove
London W2



PC CONQUEROR Q

Mike Lloyd answers questions about the Digital Precision PC emulator



Q. Instead of buying an expensive PC, can I use PC Conqueror and my QL to run MS-DOS software?

A. Yes you can, because PC Conqueror is a powerful PC emulator which can cope with most programs available on the PC market. However, there are some practical limitations which should be considered before making your decision.

PC Conqueror is a program which convinces MS-DOS that it is running on an Intel 8088 central processor (the processor family used by all PCs) when it is actually running on an entirely different cpu from the Motorola M68000 family. This is a remarkable feat, but it is only accomplished at a speed which is somewhat less than that of a low-specification PC computer.

Additionally, there are a few MS-DOS programs which do things which are not yet supported by PC-Conqueror. Version 6 of Central Point's *PC Tools* program is a case in point. Standard packages such as *Plan Perfect* and *Microsoft Word* should not have any problems running on the QL.

It is worth pointing out the obvious: without a hard disk drive and considerable extra memory a QL is not going to be able to emulate a PC with a hard disk and 640K of DOS memory.

Realistically, it is probably asking too much of your QL and of Conqueror to think that between them they could save you the cost of a PC for regular and frequent use. The QL is too slow and Conqueror is performing too complex a task for the combination to offer a serious competition to a true PC machine. If it is any consolation, similar PC emulators for the Atari 520 and the Apple Macintosh computers are also slow.

Conqueror is of most value to people who (for instance) use MS-DOS computers at work and need to carry on working in DOS when they arrive at home. Others might value learning about the the DOS environment on the QL before plunging into the PC market. Open University students who must have access to an MS-DOS computer for part of their course are buying PC Conqueror. If you already own a QL with at least one disk drive and expanded memory, the additional cost of PC Conqueror is well worth considering. If you are planning to build up a suitable system from scratch the total cost will not be justified if all you want to do is run DOS programs.

PC Conqueror is much faster on the Atari 520 QL emulator (where an Atari is convinced it is a QL which is convinced it is a PC. . .) and on the Thor derivative of the QL. There is the chance that a hardware manufacturer will market a fast replacement for the QL cpu. Any increase in the cpu's performance is going to benefit PC Conqueror quite substantially, thus making it more practical to use.

Q. Can you recommend a word processor to use with PC Conqueror?

If you are a competent typist used to the complexities of *WordStar*, *WordPerfect* and so on you will find the same programs running under PC Conqueror quite slow. You will also need to be selective in the number of program files you put on the "system" floppy disk if you do not have a hard disk fitted to your QL. It is also beneficial to use Conqueror's "supervisor screen" to increase the priority of the screen-updating program. This ensures that the screen keeps up to date with moderately fast typing.

Speed is less an issue for "hunt-and-peck" typists, who might also prefer a less dauntingly complicated wordprocessor than Word Perfect or Word. Typists who complained that the jerky cursor movements of Digital Precision's *Solution* were offputting will be impressed by the smooth and regular screen updating which PC Conqueror provides.

For most people, the choice of wordprocessor will be the same as that they are used to on the PC they use at work. PC Conqueror includes a very useful utility called "Xover" which transfers files from DOS format to QDOS format and vice versa. You can therefore bring work home in a DOS file, transfer the file to QDOS using Xover, work on it using a QDOS wordprocessor such as *text 87* or *The Editor*, and transfer it back again to DOS format for final tidying up using the wordprocessing package used by your company.

This is not quite as easy it seems because the control codes used to produce features such as underlining and italics in one package are not going to be the same as the control codes adopted by another program. With The Editor, of course, there is some latitude for tuning it to emulate the DOS program which will eventually handle the text files it produces.

With other QDOS-based wordprocessors it is quite workable to avoid all

AND A



control codes while drafting the document and insert them after the text has been transferred to the DOS package following its conversion using Xover.

If you particularly want to own a DOS wordprocessor but do not wish to spend a great deal of money on one, there are some excellent shareware programs which can be obtained for a few pounds.

Q. I have loaded Word Perfect successfully but many of the function keys seem to be jumbled up. For instance, F7 ought to allow me to quit Word Perfect but instead it calls the "search" function. What can I do?

The reader who asked this question owns a Sandy keyboard with an additional numeric keyboard and ten function keys. She made the logical assumption that F7 on the keyboard would be read as F7 by the program she was using. However, the standard QL only has five function keys. On the Sandy keyboard the keys marked F6 to F10 are actually used to represent Shift-F1 to Shift-F5. Therefore pressing F7 on the Sandy keyboard is the same as pressing Shift-F2 on a PC keyboard.

Which begs the question, where is the real F7? Steve Sutton, the programmer who devised PC Conqueror, had a difficult problem on his hands because the normal QL keyboard has considerably fewer keys than the 102-key PC-AT keyboard which MS-DOS programs assume to exist. Sutton could not make the assumption that all PC Conqueror users would use the Sandy, Schon or other replacement keyboards and so he had to devise a solution which was workable on standard QL keyboards.

By clever mapping of most of the expected DOS keypresses to sensible equivalents available to the QL, Sutton has made it quite easy to use all the DOS keystrokes. For example, the "missing" function keys F6 to F10 are found by pressing Ctrl-6 to Ctrl-10, and so the F7 which Word Perfect requires before entering the exit sequence if obtained by pressing Ctrl-7.

As the PC Conqueror manual states, the default key mapping the file can be improved upon for QL owners with additional keyboards, but it is left to them two work out the best method of rearranging the keypresses to suit their own particular needs. PC Conqueror includes a configuration utility which simplifies the construction of an alternative keymapping file.

A quick improvement can be obtained by specifying that Shift 1F- to Shift-F5 on the QL (ie the F6 to F10 keys on the Sandy keyboard) become F6 to F10 within PC Conqueror. Ctrl-1 to Ctrl-0 can be used to represent Shift-F1 to Shift-F10 in MS-DOS programs. Further refinement depends upon the selection of extended keypresses (ie those involving Shift, Alt and Ctrl) which feature in the MS-DOS software you intend to use on the QL.

Q. I have tried to use Psion Xchange's task switching feature but without success. Why does the QL not support this?

A. Actually, the QL and PC-Conqueror do support this feature, which is found on quite a few MS-DOS programs apart from the Xchange suite. To call it "task switching", however, is misleading. What actually happens is that at the moment that the user selects to switch from one Psion program to another the computer records a "snapshot" of the current state of the current application and saves it on a disk. When it is time to restore the interrupted program the contents of the application file plus the program which created it must be restored to the computer's ram. This is incredibly crude compared with the sophistication of the QL's genuine multi-tasking environment and is one major reason why my 386 PC regularly sits silently while I use my QL.

The reader's problem was that the disk placed in the "A" drive had insufficient space to hold the snapshot of his current application so Xchange could not start up another program without losing all record of the first one. This it sportingly refused to do.

The solution is to create a disk containing all of the program overlays and nothing else. If you can live without them, exclude the *. HOB help files. The disk will then have plenty of room to store a number of suspended applications. More permanent data files can then be stored on other disks. It is then simply a question of remembering to swap disks prior to loading or saving a data file.

Q. The DOS network I use at work suffered from a virus. Will my QL be affected if I bring home possibly contaminated spreadsheet files?

A. Computers with hard disks are most vulnerable to virus programs because hard disks are where viruses hide to avoid being destroyed by a system reset. In fact, a QL without a hard disk is an ideal place to test any suspicious MS-DOS disks for the presence of a virus because you can be absolutely sure that the contamination will go no further.

In order to test for a virus you will need a specially-written virus-detection program, versions of which are now in the public domain and thus can be obtained very cheaply. Only insert the disk being tested before resetting the QL to destroy any virus which might be lurking in its ram. One thing you can be certain of: there is no PC virus which can exist in the QDOS environment.

Q. I want to use *Autoroute* on the QL, but it exceeds one megabyte in size. Can it be done?

A. Autoroute is a program which selects road routes from one town to another. Like most DOS software it is not one file but many, so the trick is to select those files which are essential and discard those which are not in order to fit the program onto a single disk. The essential Autoroute datafiles cannot all be fitted onto a single 720Kb floppy disk. By splitting the files which make up Autoroute onto two disks, however, it is perfectly feasible to run the program on the QL. For a two-disk set-up, simply place both Autoroute disks in the drives and start the program from the drive containing the disk which has the batch file called "route" on it. For a single-disk system, place the disk with "route" on it in the drive and wait until the screen shows a prompt "Place Disk 1 in Drive B". At this stage, replace the Autoroute boot disk with the second disk and press any key to continue. But be warned, David Batty of Sector Software reports that a QL needs the largest Trumpcard expansion available in order to run Autoroute.

Readers may wish to write in with their tips for using Conqueror, including listings for keymapping files appropriate to different keyboards or different Dos programs. Address your letters to the Editor, Sinclair QL World using the address at the front of the magazine. Listings should be printed in clear, dark type.

ARCHIVE POWER IV

Robin Stevenson raids earlier episodes and adds some new routines.

For the last of my articles on the potential of the *Archive* programming language, a slightly different approach will be taken from previously. None of the new procedures will be directly portable, in the way the other toolkit procedures are. This is because a rather more complex task is to be accomplished. However, the core procedures are specifically designed to be as independent as possible from the particular application, so that they can be readily re-applied to other similar situations.

The problem we shall be tackling is the list. The requirement will be familiar to many users. You want to be able to browse down a list of entry lines, with a reasonable screen-full in view, but with a variable number of entries, which can be scrolled down or up when you reach the top or bottom of the screen. The Archive program editor is one good example of the genre, but it is also a regular requirement of a database file. Finding the correct record is very much easier this way than if you can only see one record at a time. And if you need to check through a number of records, altering cerain ones, a list provides a very much more natural way to do it.

Unfortunately getting Archive to handle records in this way is far from straightforward. Rather like a dog walking on its hind legs, it can do it, but it doesn't come naturally. The solution offered here is something of a compromise; a trade off between speed of use, compactness of code, and features provided. If you apply it to a requirement of your own you may need to alter that balance, to suit the job in hand.

The application is a composite to-do list and appointments diary. This is not quite such a strange union as first may be thought. The idea is that entries can be either dated (appointments) or undated — things to do, for which you have no specific date. In use, you will get a list of the entries for a particular day, and if the day you look at is today (as defined by the system clock) you will also get the list of the undated entries, in order of priority. You can readily make a mental or physical plan for the day. It could make you more organised, efficient, dynamic — well, you never know.

The various elements needed for a working list can be grouped into three heads — data, display, and control. If we look at control first, the guaranteed needs will be to step up and down the list, and to exit from it. (Other actions will depend on

Listing One. Add this to the existing toolkit procedures, saving it as a new program called TODOLIST. (See text for further details.)

```
proc LISTINIT                          : rem INITIALISE LIST VARIABLES
  let LISTWINDOW$=chr(20)+chr(2)+chr(5)+chr(78)+chr(20)
  let TEXTWINDOW$=chr(20)+chr(0)+chr(20)+chr(80)+chr(24)
  let CURRENT$=date(0): let REDRAW=2
  endproc
-------------------------------------------------------------------------
proc LIST        : rem MAIN CONTROL PROCEDURE FOR ANY LIST PROGRAM
  local KEY$,MAX,LNUM,MOVED,LOOP,COLOUR,L$
  LISTINIT
  let KEY$="E"+chr(2)+chr(3)+"AUD": let MAX=14
  let LOOP=1: while LOOP          : rem ******** START OF MAIN LOOP
    if REDRAW:LISTDRAW: let REDRAW=0: let LNUM=0 : rem DRAW SCREEN
      else :LISTLINE;LNUM: print ANSWER$;: endif      : rem OR LINE
    let L$=ANSWER$: let MOVED=0: let COLOUR=3+(DATE$<>"")
    let LOOP=instr(KEY$,inkey()): if LOOP: let LOOP=LOOP-1
      else :LISTTEXT;KEY$: let LOOP=ANSWER: endif : rem GET OPTION
    if LOOP>2:LISTOPTIONS;LOOP: endif     : rem DEAL WITH SELECTION
    if LOOP=1 and recnum()>0: let MOVED=-1: back : endif
    if LOOP=2 and recnum()+1<count(): let MOVED=1: next : endif
    print LISTWINDOW$; ink COLOUR;L$;       : rem LOWLIGHT OLD LINE
    if LNUM+MOVED<0: print chr(22)+chr(1);: else : rem SCROLL DOWN
      if LNUM+MOVED>MAX: print chr(21)+chr(1);      : rem SCROLL UP
        else : let LNUM=LNUM+MOVED: endif : endif   : rem MOVE LINE
    endwhile                          : rem ********* END OF MAIN LOOP
  endproc
-------------------------------------------------------------------------
proc LISTTEXT;K$                          : rem DISPLAY TEXT AND MENU
  local T$: print TEXTWINDOW$: screen : rem WINDOW ON LAST 4 LINES
  if count(): sprint : endif                 : rem SHOW CURRENT ENTRY
  let T$=chr(190)+chr(191)                   : rem MAKE MENU REQUEST
  GETCHOICE;"Exit  "+T$+"  Actions  Undo/Do  Dates",5,K$
  endproc
-------------------------------------------------------------------------
proc LISTLINE;LINE             : rem MAKE SINGLE LINE FOR LIST ENTRY
  if count()>0                          : rem IF THE LIST IS NOT EMPTY
    let ANSWER$=chr(25)+chr(3)+PRIORITY$+chr(9)+chr(1)+":"
    let ANSWER$=chr(31)+chr(0)+chr(LINE)+ANSWER$
    let ANSWER$=ANSWER$+TEXT$+chr(9)+chr(75)+DONE$
    else : let ANSWER$="": endif
  endproc
-------------------------------------------------------------------------
proc LISTDRAW                         : rem DRAW FULL LIST FROM SCRATCH
  DATEFORMAT;CURRENT$: print HEADING$
  if CURRENT$=date(0): let ANSWER$=ANSWER$+" **TODAY**": endif
  print at 4,0; ink 0; paper 4;"  ";ANSWER$; tab 75;LISTWINDOW$
  cls : if REDRAW=2: reset : order DAYS;D,PRIORITY$;A
    if CURRENT$<=date(0): select DAYS<=days(CURRENT$)
      else : select DAYS=days(CURRENT$): endif : endif
  last : while recnum()>0             : rem LOOP TO PRINT EACH LINE
    LISTLINE;0: print ink 3+(DATE$<>"");ANSWER$;chr(22)+chr(1);
    back : endwhile                          : rem END OF PRINT LOOP
  LISTLINE;0: print ANSWER$;                    : rem PRINT TOP LINE
  endproc
-------------------------------------------------------------------------
proc LISTOPTIONS;OPT      : rem HANDLE OPTIONS 3,4 & 5 OF MAIN MENU
```

```
  local T$,P: if OPT=3: error TODOACT: return : endif
  if OPT=4: let DONE$="-X"((DONE$="-")+1): update : else
    let T$="Exit  Next  Previous  Forward"       : rem DATE OPTIONS
    GETCHOICE;T$+"  Back  Date  Today",6,"ENPFBDT"
    if ANSWER=0: return : endif : let T$=CURRENT$: let P=recnum()
    if ANSWER<3: let YES=1: reset : order DAYS;D,PRIORITY$;A
      locate days(CURRENT$)-(ANSWER=2)          : rem PREVIOUS ENTRY
      if ANSWER=1: back : endif                    : rem NEXT ENTRY
      if DATE$<>"": let ANSWER$=DATE$           : IF NO ENTRY FOUND
        else : let ANSWER$=date(0): endif : endif : rem GOTO TODAY
    if ANSWER=3 or ANSWER=4           : rem FORWARD OR BACK ONE DAY
      DATE;days(CURRENT$)+(ANSWER=3)-(ANSWER=4): let YES=1: endif
    if ANSWER=5                     : rem ENTER A GIVEN DATE TO GO TO
      GETSTRING;"Enter Required date",""
      error DATECHECK;ANSWER$: endif          : rem CHECK IT'S VALID
    if ANSWER=6: let ANSWER$=date(0): let YES=1: endif
    if YES: let CURRENT$=ANSWER$: endif  : rem UPDATE CURRENT DATE
    if T$<>CURRENT$: let REDRAW=2: else : position P: endif
    endif
  endproc
-------------------------------------------------------------------
proc TODOLIST                       : rem STARTS THE TODOLIST PROGRAM
  error TODOCLEAR: screen : sload "TODOLIST"   : rem FILE & SCREEN
  error LIST                              : rem RUN THE LIST PROCEDURE
  error SHUT;""                              : rem CHECK ALL IS TIDY
  endproc
-------------------------------------------------------------------
proc TODOFILE;M$,F$         : rem OPEN OR CREATE REQUIRED LIST FILE
  local T$: error FOPEN;M$,F$            : rem FIRST TRY TO OPEN IT
  if YES: return : endif
  let T$=ANSWER$+" not found : "      : rem OFFER TO CREATE NEW ONE
  GETCHOICE;T$+"0. Abandon  1. Retry  2. Create file",2,"ARC"
  if ANSWER=0: error : endif                  : rem ERROR TO ABANDON
  if ANSWER$=1:TODOFILE;M$,F$: return : endif          : rem RETRY
  create ANSWER$ logical F$                  : rem OR CREATE NEW ONE
    DATE$                                 : rem OPTIONAL DATED ENTRY
    DAYS                               : rem NUMERIC DATE, FOR ORDERING
    PRIORITY$                       : rem OPTIONAL PRIORITY FOR ORDERING
    DONE$                                 : rem FLAG TO MARK AS DONE
    TEXT$                                 : rem MAIN LIST TEXT FIELD
    endcreate
  endproc
-------------------------------------------------------------------
proc TODOCLEAR          : rem GET TODOLIST FILE, & CLEAR OLD ENTRIES
  TODOFILE;"Enter filename for To Do List","TODOLIST"
  let TFILE$=ANSWER$: order DAYS;D,PRIORITY$;A  : rem ENSURE ORDER
  TODOFILE;"File Is "+TFILE$+"_HST","HIST"     : rem GET HIST. FILE
  CENPRINT;2,"CLEARING FILE": use "TODOLIST": last
  while DATE$<date(0) and recnum()>0 and lower(inkey())<>"q"
    if DAYS>0 or DONE$="X"             : rem IF OUT OF DATE, OR DONE
      let HIST.DATE$=DATE$: let HIST.DAYS=DAYS  : rem COPY ACCROSS
      let HIST.PRIORITY$=PRIORITY$: let HIST.TEXT$=TEXT$
      let HIST.DONE$=date(0): append "HIST"        : rem AND APPEND
      use "TODOLIST": delete           : rem BEFORE DELETING OLD ONE
      else : back : endif
    endwhile
  error SHUT;"HIST": use "TODOLIST"
  endproc
-------------------------------------------------------------------
proc TODOACT                  : rem PROCESS THE TODO ACTIONS REQUEST
  local T$: let T$="0. Exit  1. File  2. Insert"
  GETCHOICE;T$+"  3. Alter  4. Print  5. Delete",5,"EFIAPD"
  let C=ANSWER
  if C=1:TODOCLEAR: endif     : rem CHANGE TO A DIFFERENT LIST FILE
  if C=2                            : rem INITIALISE A NEW LIST ENTRY
    let DATE$="": let DAYS=0: let PRIORITY$="": let DONE$="-"
    let TEXT$="": let REDRAW=1: append : endif
  if C=2 or (C=3 and count()): print HEADING$; : rem INSERT/ ALTER
```

the particular application.) The data can be similarly straightforward — the list can be any Archive file. By using the SELECT statement, a part of a file can be used. One selected it behaves exactly like an entire file.

As far as the list is concerned the whole file is available to it, while each application must define which fields are to be used in the one line display of data. The list will have to be drawn completely in a number of circumstances: at the start; whenever a different list is selected; and if the screen is used for something else. Individual lines will need to be highlighted, to show the current line, and if you move beyond the top or bottom of the screen, it must be scrolled accordingly, before writing the new line.

The procedures in **Listing One** perform three various functions for the To-do list. They can be divided into two types. Firstly, there are those called LIST something. These would all be needed in some form (modified to suit the application) by any list. They contain most of the elements mentioned above. Then there are those called TODO something. These perform tasks much more specific to the To-do list. They may well have direct parallels in other applications — file creation, printing, etc. — but they are not essential to the list concept.

The listing does not provide a complete program in itself. It builds on a number of toolkit procedures developed over the last three months. Unfortunately the required procedures are spread over the NOTEPAD and CALENDAR programs, so unless you have bags of memory, and are adding all the programs in together, you will need to extract them to create a new TODOLIST program. From the NOTEPAD program you will need CALCSETUP, CALCULATOR, CENPRINT, CONFIRM, FOPEN, GETCHOICE, GETSINPUT, GETSTRING, PRINTOFF, SHUT, SPOL, and WINDOW. Delete all the other Notepad procedures, and then save those procedures as TODOLIST. To these, you will need to add the three date handling procedures from CALENDAR. These are DATE, DATECHECK, and DATEFORMAT. Delete all the other calendar functions and then merge in the other procedures by typing in the command MERGE "TOODOLIST". The file can then be again saved as TODOLIST, and is ready to have the LIST procedures added to it.

Users of last month's calendar program will have noticed a reliance on the QL system clock, to determine today's date. This dependence is even greater in the To Do list. If the program is to automatically prune out old appointments, and thinks the year is 2027 or some such, you will find you have a remarkably empty appointments book. Unfortunately there is no provision within Archive for altering the clock. It must be set from SuperBasic before Archive is called. Unless you happen to have a fancy battery backed clock in your QL, the best way to do this is

```
    CENPRINT;2,"Enter new Date, Priority, and Text."
    print TEXTWINDOW$;: screen : sprint
    let T$=CURRENT$: let P$=PRIORITY$
    alter : if T$<>DATE$: error DATECHECK;DATE$
      let DATE$=ANSWER$: if ANSWER$<>"": let DAYS=days(ANSWER$)
       else : let ANSWER$=date(0): endif
      update : let CURRENT$=ANSWER$: endif
    if P$<>PRIORITY$: let REDRAW=1: endif  : rem IS THE ORDER DIFF
    if T$<>CURRENT$: let REDRAW=2: endif : endif : rem OR THE DATE
  if C=4:TODOPRINT
    error FOPEN;"File Is "+TFILE$,"TODOLIST": error PRINTOFF
    let REDRAW=2: endif
  if C=5 and count(): delete : let REDRAW=1: endif
  endproc
-------------------------------------------------------------------
proc TODOPRINT
  local D$,X,INDENT: let D$=TFILE$
  GETCHOICE;"PRINT : 1. Old  2. Current  3. Todays",3,"FOCT"
   if ANSWER=1: let D$=D$+"_HST": endif
   error FOPEN;"File Is "+D$,"TODOLIST"    : rem CHECK FILE IS OPEN
   order DAYS;D,PRIORITY$;A                : rem CHECK IT IS IN ORDER
   if ANSWER=3: locate days(date(0)): endif    : rem START AT TODAY
   SPOOL: let D$="": while not eof()   : rem ****** MAIN PRINT LOOP
     if D$<>DATE$: if DAYS=0: let ANSWER$="UNDATED ENTRIES"
        else :DATEFORMAT;DATE$: endif : lprint   : rem SUB-HEADING
      lprint tab MARGIN;ANSWER$: let D$=DATE$: endif
     let INDENT=2: while len(TEXT$)>60: let X=60  : rem FORMAT LOOP
       while not instr(" ,.-):;",TEXT$(X)): let X=X-1: endwhile
       lprint tab MARGIN+INDENT;TEXT$( to X): let INDENT=4
       let TEXT$=TEXT$(X+1 to ): endwhile  : rem END OF FORMAT LOOP
     lprint tab MARGIN+INDENT;TEXT$
     if len(DONE$)=10
       lprint tab MARGIN+40;"Archived : ";DONE$: endif
     next : endwhile           : rem ******* END OF MAIN PRINT LOOP
  endproc
-------------------------------------------------------------------
proc START              : rem LINK IT IN WITH MAIN DESKTOP PROGRAM
  error TODOLIST                          : rem CALL THE APPLICATION
  run object "NOTEPAD"                     : rem & RETURN TO NOTEPAD
  endproc
```

to incorporate a routine into the Archive BOOT file. You will need to renumber it, so that extra lines can be inserted before the call to exec Archive. For those who have not yet written their own, I offer the following

```
.....
50 INPUT 'Enter day : ';d$,'Month :
   ';m$,'Year : 'y$,'
60 INPUT 'Time Hours: ';h$,'Minutes :
   ';mn$
70 IF LEN(y$)=2:y$='19'+y$
80 SDATE 0&y$,0&m$,0&d$,0&h$,
0&mn$,0
.....
```

This has the advantage that if you are sure you won't be needing it, you can flick past with five rapid Enters. While dealing with BOOT, I shall repeat something discussed in an earlier article. If you have found insufficient memory for these programs, replace the exec_w line in the boot program, with NEW. This will clear out all superbasic and variables (but won't reset the clock), and give you a couple more kilobytes. You will then need to type manually the instruction EXEC_W MDV1_ARCHIVE.

We shall no look at the general list procedures, starting with LISTINIT. This initialises four new global variables, for use by LIST. Two of these are in effect constants, for sizing the two windows used. Storing them as string variables in this way allows greater speed than calling the WINDOW toolkit procedure, and also makes it easier to change the window size than having to find each entry in the program. Remember also that the HEADINGS constant, initialised by the DESKTOP initialisation program, also contains a window size — the full screen area — so LIST can quickly use any of these three window areas. There are also two true global variables in the LISTINIT procedure — REDRAW, and CURRENTS. Redraw is a flag to show whether the screen needs redrawing or not, and Currents is used to control the selection of the required records. In our case it holds the date currently being looked at.

LIST is the main control procedure for the program. It controls list display, menu requests, movement around the list, and access to other options. The only one of these it handles directly is movement around the list. All of the others are subcontracted to other procedures, making LIST a concise and hopefully understandable procedure. The four elements are dealt with by the procedure in turn.

LIST

First, the screen display. If the REDRAW flag has been set, LISTDRAW redraws the entire screen, otherwise LISTLINE highlights the current line. Next the keyboard is checked. If a valid response is already entered, it is acted on directly, otherwise LISTTEXT provides a full menu. Much faster movement around the list is possible than if the response was entirely in the hands of LISTTEXT. If the request is not one of Up, Down, or Exit, it is simply passed on to LISTOPTIONS. Movement up or down the list requires checking that there is another record to move to, and then whether or not the screen needs scrolling or the line number moving.

Seven local variables are required to service these functions. The first two are constant which help with portability — KEY$ is a string of the allowable input letters, and should match the initials used in the LISTTEXT menu. MAX is the maximum number of lines available in the LISTWINDOW$ window, so any changes in the window size would need a comparable change to MAX. The next three are control variables — LNUM for the current screen line number, MOVED for the direction moved along the list, and LOOP for the user response and WHILE LOOP control.

Portability

Finally, there are two display line variables — L$ stores the highlighted line, so that once another line is found, it can be 'lowlighted' before highlighting the new one. Finally COLOUR is the one application specific feature of the procedure. Date entries are to be green, and undated entries red in the To Do list, with the highlighted line white. COLOUR stores the colour to return it to, after moving on.

And so on to the LIST support procedures. There are four of these. They all perform functions essential to any list, but the details of each will be specific to the particular application.

LISTTEXT;<s.exp>

Whenever nothing else is being done with the list, it uses its time providing helpful information to the user. This falls into two parts — information about the current line, and a menu of the available options. The first of these is a standard Archive screen, placed in a window at the bottom. **Figure One** shows the screen you will require. It is also used as the data entry screen, when inserting or amending items. The menu is then provided via the toolkit procedure — GETCHOICE. To ensure agreement between the keypress

options of LISTTEXT and LIST, the KEY$ variable is passed as a parameter to LISTTEXT, and from there to GETCHOICE.

GETCHOICE itself may require a second look here. The problem is the online calculator, which relies on a SEDIT screen variable called ANSWER$ being in the menu area. TODOLIST uses a smaller screen, windowed at the bottom of the QL screen, so cannot provide this sèrvice. To make the calculator usable you will need to customise GETCHOICE so that it loads the NOTEPAD screen before running the calculator, and reloads the TODOLIST screen after use. Find the line the call CALCULATOR, and add the two comamnds, so that it reads 'sload "NOTEPAD": error CALCULATOR: sload "TODOLIST".'

LISTLINE; ‹n.exp›

One of the benefits of the Archive screen driver is the ability to combine printed material and control instructions in a single string variable, for use elsewhere. LISTLINE makes use of this to write a line of the list. It is needed by both the main LIST procedure, and when drawing a list from scratch. In particular it needs to be highlighted, when it is the current line, and lowlighted again afterwards. LISTLINE provides the line of text, with embedded start point (provided as a parameter) and tabulation instructions, to give a tidy, organised list. An additional screen driver command (CHR(25)) is used to prevent long entries from wrapping on to the next line.

LISTDRAW

As the name suggests, this procedure draws the list in full, when required by the program. However it also performs another equally important task — selecting the appropriate records from the file. The REDRAW flag causes LISTDRAW to be run whenever it is not zero. However if it is set to two, this denotes not only that the dislay needs redrawing, but that a different selection of records is required. The global variable CURRENTS is used to select records relating to the day in question, and also any undated records, if required. Once selection is complete, the entire current list is printed in reverse order, so that it scrolls down from the top of the screen. This is a little cumbersome for a long list, but has the dual advantage of being easy to code, and gives a glimpse of the bottom of the list, before it scrolls off the end. The end of the list is the low priority entries, but you don't want to forget them completely.

LISTOPTIONS;‹n.exp›

The last of the LIST something procedures handles the top level menu options not already provided for in LIST. These are all application specific, but will need replacing with something in all except the most simple list applications. In our case, the Actions request is passed on to the TODOACT sub-menu. The Undo/Done option simply toggles a flag to tick off an item as done, or re-instate it if it wasn't done properly! The remainder of the procedure is a sub-menu to change to a different date. Six options are offered. You can go to the next or previous day which has an entry; you can go forward or backward one day; you can specify a particular date to go to; or you can go directly to today's date. Throughout,

```
@RJS                         ARCHIVE DESKTOP                          1990
                                    3
                     Enter new Date, Priority, and Text.

 Fri  30 Mar 1990 **TODAY**
| 1:11.30  - print SQLW examples.                                      |
| 2:2.00 - John Smith : Ilford Mortgage and Investment Centre - interested in-|
| 3:Evening - visit Parents.                                          ~|
| A:Finish debugging todolist                                          |
| A:Redraft Article 4, SQLW                                            |
| D:Repair Car petrol tank (jubilee clip)                             ~|

DATE : 1990/03/30  (FORMAT YYYY/MM/DD)        PRIORITY   2    (A-Z)
TEXT : 2.00 - John Smith : Ilford Mortgage and Investment Centre - interested i
       n a client database.  57 Accacia Avenue, Ilford. (01-345 6789)
L_________BYTES :   3460__________________Fri  30 Mar 1990  11:40:40__________J
```

```
| HELP       | LAYOUT mode                              | COMMANDS  |
|   press F1 | F3 to set variable                       |  press F3 |
| PROMPTS    | Use cursor keys and ENTER to move        | ESCAPE    |
|   press F2 | cursor and type screen background        | press ESC |

DATE : ..........  (FORMAT YYYY/MM/DD)        PRIORITY   ..  (A-Z)
TEXT : ......................................................................
       ......................................................................

      The 3 variables are DATE$, PRIORITY$, AND TEXT$.
      The double line of TEXT$ is obtained by extending the first line of dots
      to the edge, using the right-arrow, and then pressing the down-arrow once.
      This area, below the TEXT$ dots should be left blank.
```

0 ←Line of red tildes ('~')
1 Variables 'Dates' and 'priorities'
2
3 } Two line 'Text$' variable

(Only the top four lines are used)

Version 2.3 only

ANSWER$ temporarily holds the new date, and YES marks it as successfully found. The REDRAW flag is only reset if the new date is different from the old one.

TODOLIST

This provides the start point for the To-do list, but hands over full control to the more general LIST procedure. May of the features needed are incorporated into LIST already. However we are still some way from a workable program. We still have no data file, no means of entering or modifying data, nothing to clear out old entries, and no way of printing out the lists. These actions are all very specific to this application, and are covered by the remaining TODO procedures. We shall look first at the data file.

TODOFILE;<s.exp>,<s.exp>

This provides a more specific front end to the toolkit FOPEN procedure, being passed a message, and logical file name as parameters. If FOPEN fails to find the named file, TODOFILE gives the user a choice: abandon the attempt; try again; or create a suitable file of the name given. The default file name is TODOLIST, and this would normally be acceptable, but if you require compartmentalised lists, or if it is used by several people, each could use a separate file.

If you don't abandon the attempt (which forces an error), and you don't re-try (which calls TODOFILE recursively), a new file will be created. The data structure is fairly simple. There are direct text fields where you can enter the date (if required), a priority value from A to Z, and a description of the job to be done, or appointment to be kept. The other two are hidden fields, one holding the flag to mark a job as done, and one to store a numeric value for the date. This last is necessary because Archive only orders the first eight letters of a text field.

Since the date format uses ten letters, the all important day of the month would be ignored when locating a date. To avoid this, every new entry with a date must use the DAYS() function, to store the numeric value as well, and it is this DAYS field which will be used as the primary ordering for the file. As a secondary ordering, PRIORITY$ is used so that low priority entries end up lower down the list.

TODOCLEAR

Eash list will have two files for its use, with an identical field structure. One will have a normal '_DBF' suffix, and store the current and future entries. The other will have the same name, but an '_HST' suffix to the name, and will store the historical records, cleared out from the main file. Using non-standard suffixes, with the same stem name, is a very handy way of showing the link between related files. It will be useful at this point to look at the sequence of events for starting a To Do list session.

Before running LIST, the TODOCLEAR procedure is run. (Exactly the same code is used if you change to a different file during the session.) First the required file is obtained, and stored in the global variable TFILES, using TODOFILE. Next the historical counterpart for this is obtained in the same way. Starting at the end of the current file, each record is checked. If it is marked as done, or it it is past its do-by date, it will be transferred to the historical file. Once today's entries are reached, the search stops. Future records, even if marked as done, will not be reached so will not be transferred. When complete, the historical file is closed, so that LIST will deal with the correct set of data. When the _HST file gets too large you could copy it to a different drive if it may be needed again. The original can then be deleted, and a new one will be created next time you run the to-do list.

TODOACT

The Actions sub-menu is controlled from the TODOACT procedure. The File and Print requests are simply passed on the TODOCLEAR or TODOPRINT procedures, while the record handling options, Insert, Alter and Delete are handled in-house. The first two of these are identical, except that Insert has to clear and append a new record before it can be 'altered' with new data. That done, the task of displaying prompts, restoring the active screen, using Alter to allow editing, and checking for date validity are all common to both.

In all three cases, complete screen redrawing may be necessary, and is the most unsatisfactory aspect of the LIST. The alternative, handling the deleting and inserting of individual lines in the list, would only be feasible on a expanded QL, as it would require significant extensions to the program.

For anyone feeling adventurous this would make an interesting project. You would need to keep track of the records at the top and bottom of the window, and utilise the partial window scrolling provided by screen driver chr(27). You will also have to find a way of discovering where in the list the new/altered records has been moved to. And if the date has been changed, the screen will need redrawing anyway. If you still think it is worth it, perhaps Open Channel would be interested in your efforts. No prizes I'm afraid.

TODOPRINT

The last procedure covered controls the printed output. Three different options are offered — to print all of the historial records (the only thing you can do with them), print all of the current records; or print the list of todays/undated entries. The same print redirection options are used as for the notepad, (printer, screen, or file) and the list is then printed out, formatted to fit an eighty column page.

Two programming points of interest are used, which will be common needs for many data output applications. These are sub-headings and text formatting. Sub-headings are the more straightforward of the two. The file must be ordered, and the ordering field (often called a 'key' field) provides both the control and the content of the sub-heading. In our case DATE$ is the control variable, to give a list of all the entries for each day under that day's sub-heading. Each time round the loop DATE$ is compared with the stored variable, D$. Whenever they are not identical, a new sub-heading is printed, and D$ updated to the new date.

Formatting each line of text is a little more complex, involving two nested loops, however the end result is so much better than simply chopping words in half that it is well worth it. The problem is that we have generously allotted over 140 characters to the text line. In many cases only 40 or 50 of these may be used, but where a long entry is needed, legible output is needed. The first WHILE condition is used if the line is longer than 60 characters. If it is not, then the entire loop is ignored, and the final print line will print out the line.

Separator

If the line is longer, the inner loop is entered. Starting at position 60, the line is scanned backwards until a separator is found (defined here as a space, comma, full-stop or hyphen). This being the first safe place to split the line, it is printed up to that point, and removed from the text variable. (Because no UPDATE is used, the original is unaffected.)

The same process will again take place, until less than 60 characters remain, at which point it exits the outer loop, and the final line is printed by the final print instruction, as for lines that start short. The INDENT variable addes a slight embellishment to his. Before entering the loop it is set to two, so the first line to be printed will tab in by the margin plus two, regardles of which print instruction is used. If the loop is entered, INDENT is set to four, so that all subsequent lines, including the final one will indent four spaces. If the application required it, INDENT could be set for less rather than more, after the first line, so that the first line is indented, rather than the subsequent ones.

All that remains, once the To Do list is fully entered, debugged, and saved is to add the START link procedure — and re-save it. It can then be run automatically by the NOTEPAD control program, and to find its way back there afterwards.

I hope that you have found something of use in this series. We have covered a range of programming techniques, and application solutions, which could be readily adapted to other situations. We have also provided a number of 'toolkit' procedures, which can be placed directly into other programs, to handle many of the things common to all situations. As a result you should find it worthwhile to run up quick, effective programs for all sorts of problems which may have seemed out of reach until now.

BOOK PAGE

Mike Lloyd examines two books which could improve programming skills.

A Concise Advanced User's Guide to MS-DOS
By N. Kantaris
Published by Bernard Babani (publishing) Ltd., The Grampians, Shepherd's Bush, London W6 7NR.
Price £2.95.

This slim volume

Not very long ago every newsagent's magazine shelves were under threat from a tide of computer titles. As that wave receded it was the turn of the biggest high street booksellers to suffer from the computer invasion. As I know to my cost, with a 50,000 word manuscript still unpublished, this surge died even more quickly than the first.

Huge

Recently, though, there seems to have been a turn in the fortunes of the third party computer manual dedicated to the MS-DOS operating system and the major pieces of software which run under it.

Rather in the manner of pulp novels sold, as legend has it, by the pound, the new wave of computer titles are monolithically huge. Being mostly of American origin, they also tend to be impenetrable. As an antidote to such gigantism I have been reading one of the slim budget volumes produced by Barnard Babani (publishing) Ltd of London.

Interest

About the only large attribute of Noel Kantaris's *Concise Advanced User's Guide to MS-DOS* is the title. However, in less than 70 pages the author delves into MS-DOS batch files, the mysteries of the ANSI.SYS commands and the complexities of debugging assembler routines. These are the very topics of interest to all QL owners have have invested in Digital Precision's marvellous Conqueror MS-DOS emulator.

The book is liberally decked with useful reference tables and is written in a clear style which successfully avoids patronising on the one hand and pretentious techo-speak on the other. The author is the Head of Computing at the Camborne School of Mines and has a number of other DOS-related books to his credit in the Babani catalogue.

Batch

The whole object of the book is to introduce some advanced DOS programming techniques through the medium of writing batch files to display and manage menus. It is the equivalent of introducing readers to SuperBasic by demonstrating how to write sophisticated BOOT files. This choice of approach neatly brings together the subjects of batch files, the EDLIN line editor, many MS-DOS housekeeping routines and simple assembler programs. Readers can therefore use the book as a springboard to greater knowledge or as a reference for writing relatively simple but worthwhile DOS utilities.

Pound for pound, Kantaris's book is arguable better value than its enormous rivals. Some might argue that the book's title is more ambitious than its contents, but the contents are well worth the cover price and I look forward to reading more books in the series.

Software Engineering
By I. Somerville
Published by Addison-Wesley
Price £14.95

Densely packed but never abstruse

Noboby writes programs any more; it's all "software engineering" now. The analogies between writing substantial programs and more traditional engineering skills are quite apposite and the term software engineering, coined some 20 years ago, has gained general acceptance.

Valuable

Dr Ian Sommerville of the University of Strathclyde is a respected lecturer in the subject of large-scale programming projects, and his densely-packed — but never abstruse — book of over 300 pages is a valuable reference for programmers intent on producing their multi-thousand-line masterpiece.

Applicable

Understandably, the book's perspective is from the mainframe user's standpoint, but the principles it covers are applicable to all types of programming. Topics include the software life cycle, the analysis of the user's requirements, the specification of the software, four different design strategies, programming methodology and style, a review of different languages, program testing and program documentation. Useful insights into the psychology of the computer user, the group psychology of computer programming teams and the problems of managing a software project are added to the back of the book.

Unnecessary

The University mainframe environment intrudes somewhat unnecessarily into some parts of a book which in fact appeals to a much wider computing fraternity. I would have preferred fewer references to the Ada programming language and cannot agree that languages are best categorised according to the way in which memory is allocated to data storage.

The most useful parts of the book tackled the potentially nebulous issues of program analysis and design in a down-to-earth manner. There is a closely-argued three-page analysis of the limited circumstances in which GOTOs are acceptable. The discussion of the user interface examines the attitudes of users and suggests ways of simplifying screen designs in order to make programs easier and more attractive.

Rewarding

Software Engineering is not always easy to read but it is rewarding. If you have started on an ambitious programming project which has foundered through lack of planning and organisation, this book could show you the way to success.

DIY TOOLKIT



DIY Toolkit has been a popular part of QL World for four years. This month we present updates, a review of the series, and good news for slow typists: DIY Toolkit is now available on disk and microdrive cartridge.

DIY Toolkit builds on SuperBasic and QDOS,the programs built into every QL.

In theory every QL user can customise and extend their system with personal machine-code nuggets. In practice this is a big job unless others contribute ideas and share the fruits of your effort. DIY Tookit pools ideas and share the fruits of your effort. DIY Toolkit pools ideas, technique, and information among QL users world-wide.

Simon Goodwin introduces a host of DIY Demos, and a new library of Toolkit updates and extensions.

SuperBasic is a kind of construction set, and DIY Toolkit provides components which you can use to make new things, or improve old ones.

Toolkit routines make good examples. QDOS extensions must be concise, compatible, efficient and re-entrant, so that they can be shared between several tasks

```
100 TEST : STOP
110 REMark Type RUN, WHY or TEST to see WHY in action
120 :
130 DEFine PROCedure WHY
140 LOCal rtstart,rtend,i,calltype,callline
160 REMark Show contents of RETURN stack (non-destructively)
170 REMark Type WHY to find out which calls are in progress
180 :
190 CLS 2 : PRINT : REMark Tidy remainder of window
200 rtstart=BASIC_L(56)
210 rtend=BASIC_L(60)
220 i=rtend
230 REPeat scan_rt
240   calltype=BPEEK%(i-8)
250   callline=BPEEK_W%(i-6)
260   SELect ON calltype
270    =1,2,3
280       SHOW_LINE
290       i=i-24
300    =0
310       SHOW_LINE
320       i=i-8
330   END SELect
340   IF i<=rtstart THEN EXIT scan_rt
350 END REPeat scan_rt
360 END DEFine WHY
370 :
380 DEFine PROCedure SHOW_LINE
390 IF callline=0
400   PRINT "Called by a direct SuperBASIC command."
410 ELSE
420   PRINT "Called from "; : LIST #1,callline
430 END IF
440 END DEFine SHOW_LINE
450 :
460 DEFine PROCedure TEST
470 GO SUB 540
480 END DEFine
490 :
500 DEFine PROCedure TRY2
510 WHY
520 END DEFine
530 :
540 TRY2
550 RETurn
```

```
SAMPLE OUTPUT from WHY:

Called from 510 WHY
Called from 540 TRY2
Called from 470 GO SUB 540
Called from 100 TEST : STOP
```

SuperBasic

SuperBasic was designed for the Super Spectrum, planned by Sinclair in 1982 but never released. In 1983 Jan Jones' language was converted for the QL, and bolted onto Tony Tebby's QDOS operating system.

Plain vanilla SuperBasic contains over a hundred built-in commands and functions; these are enough to write most straight forward programs. But the QL can do more than most micros, because it is extensible and supports 'big computer' features like multi-tasking and device-independence, borrowed from the Unix operating system.

Multi-tasking means that several programs can run and communicate on one machine. It can be a boon when you're busy, and means that complex programs can be built up from co-operating parts, increasing flexibility and the potential for re-use of code.

The QL rom includes the essentials for multi-tasking, but few of these facilities are available from Sinclair SuperBasic. DIY Toolkit is a bridge between the code buried in the QL rom and the applications we all use.

Device-independence means programs should work consistently with any device, real or imaginary. QL 'logical devices' are much more flexible than real physical devices, because logical devices all appear to work in much the same way. You may use several physical devices at once, as when you access a remote device over the QL network, or print via a Spooler program that temporarily buffers output.

Last year DIY Tookit illustrated all the steps involved in writing a device driver. Luca Pivato's MEM means any task can access any memory address on any machine, just as it would read or write a file. MEM allows unrestricted communication between concurrent or sequential tasks. It's also a neat example of the techniques needed to write any device driver.

```
Channels
Num   Tag   Owner          address  Name

0     0     SuperBASIC     176640   CON_448x40a32x216_136
1     1     SuperBASIC     177696   CON_448x200a32x16_136
2     2     SuperBASIC     177952   CON_448x200a32x16_136
3     33    QLTurboQuill+  178208   CON_512x256a0x0_24
4     18    Clock          235888   SCR_60x20a448x206
5     84    QLTurboQuill+  235280   SCR_448x250a34x6
6     83    QLTurboQuill+  235392   SCR_448x250a34x6
7     82    QLTurboQuill+  235504   SCR_448x250a34x6
8     81    QLTurboQuill+  235616   SCR_448x250a34x6
9     80    QLTurboQuill+  235728   SCR_448x250a34x6
10    79    QLTurboQuill+  242416   SCR_448x250a34x6
11    76    SuperBASIC     242528   N1_ram1_temp : Transput file
12    77    SuperBASIC     243296   NET
13    78    SuperBASIC     243632   MDV1_boot : Input file
14    85    QLTurboQuill+  244864   FLP1_quil_bob : Transput file
15    87    SuperBASIC     247728   SER
```

without causing clashes or hogging resources.

For instance PURGE (January 1989) is handy when tasks go awry, or you need to free lots of memory, as it gets rid of all tasks except SuperBasic. PURGE uses only 20 bytes to tell QDOS the name and address of the code.The skill comes in choosing the right 20 bytes, and typing PURGE at the correct moment. That's where the 'extras' in DIY Toolkit come in useful.

DIY routines are more comprehensively documented than commercial Toolkits, as we publish the assembly source and details of its workings as well as user notes and binary code. Most publishers keep source code secret, to 'protect' themselves from 'rip-offs' – but that contradicts part of the appeal and value of a Toolkit.

Good Toolkit code is hard to write, especially in a climate of secrecy. It's tempting to make assumptions which might be valid in a stand-alone program, but restrict the potential of Toolkit code. DIY Toolkit sets out to avoid these limitations, or document them when they are unavoidable - rather than hope that no one notices snags, leaving hundreds to learn the hard way.

The technical minutiae may pass you by today, but the details may come in useful if you need to do something out of the ordinary. You have the option to pick and use published routines, customise my assembly code, or use it as inspiration for new extensions.

This library has been finalised on a megabyte Thor XVI running *DevPac*, Metacomco's *ASM* and Psion *Xchange* on this document, alongside several other *Quill* files, with two microfloppy drives, a couple of ram disks, and a network link to a 640K Samsung QL with Greek roms, one 5.25 inch QL drive, microdrives and ram disks. I live on the QL and Thor, the way some hackers live on modems.

The assembled binary code is Public Domain. In other words, it may be freely used by commercial programs as long as the DIY part is not sold or used as a specific sales gimmick. DIY Toolkit routines have been used by major QL publishers like CGH Services, Digital Precision, Quanta, Sector Software and others, although attribution is rare. The revised documentation and assembly source, however, are protected by International Copyright Law, and may not be copied without consent.

Until now, DIY Toolkit has only been available in printed form, though *QL World*. I have mixed Basic examples and demonstrations with diagrams, text, machine code and assembly source.

Regular publication means that I can publish corrections or extra notes, if readers run into difficulty. In practice problems have been few as I try to provoke programs into failure before they are printed, testing their limits, compiling them various ways, and suffling roms to track down weak points.

DIY Toolkit files are produced on a semi-automatic production line. As a professional QL programmer from the start, and a micro writer since 1979, I have learnt that systematic design and testing are essential. I set out to eliminate problems *before* publication, and test programs on versions of QDOS from "AH" through *Argos* to *Minerva*.

When the code is cooked I use specially-written software tools to avoid transcription errors, generate the magazine listings and check that hex and assembly files correspond. Tried and tested Toolkit commands are at the core of this system.

The limited space in the magazine has good and bad side effects. It encourages me to pack a lot into a tight space, and that's a desirable attribute for a Toolkit. But it can also mean distracting cross-references to past articles, to avoid covering old ground. Often I have to condense useful examples, or leave them out altogether.

Perhaps the biggest limitation is the need for readers to type each listing if they want to take advantage of DIY Toolkit code. This can mean that busy people miss out, even though they may have much to gain from DIY Toolkit. It also penalises those who have trouble reading and keying listings accurately. I've typed and checked all the code, so why should others have to repeat this boring task?

DIY disks

Now DIY Toolkit is available on 3.5 and 5.25in disk and microdrive cartridge, by arrangement with Richard Alexander of CGH Services. The offer is a partnership between programmers Simon Goodwin and Phil Spink and publisher Richard Alexander.

Many readers have already typed in parts of the Toolkit, so the offer is split into three volumes, generally based on two magazine articles, plus updates and new programs. You don't have to buy more than you need. DIY Toolkit must pay for itself, but the costs can be low because there is no need for expensive advertising. QL World benefits because the companion disks and tapes make the articles more relevant and useful to readers.

I considered supplying the material from each month separately, as that would make my life relatively simple, but in many cases it would be annoying, as articles and demonstrations often refer to previous code or commentary. The result is a library of twelve volumes, each with a name and code letter. I have re-typed, expanded, updated, merged and edited the text for each, to remove duplication and reduce the need for cross references. Each volume can stand alone.

This is a chance for people to get hold of the assembly code for Toolkit routines they already use. It's a chance to obtain the upgrades and improvements I have made since the routines first appeared, and the extra demonstrations that outgrew the magazine format or have been written since. Finally, it's an opportunity to use QL improvements that you did not have time to enter from past issues.

Twelve Volumes

Volume B - BASIC TOOLS

This volume is a Toolkit for SuperBasic. Six routines let you look through the interpreter's own memory areas, and change things without risk of the patient moving as you POKE it. Thus you can do wonderful things, *consistently,* without crashing the machine. The volume includes four Basic access routines from December 1988, plus tested code for the two procedures set as assignments, and two new assignments for the 1990s.

Ten demonstration routines are designed to provoke thought and experimentation. A SuperBasic token disassembler reveals each component of lines in memory. It's an ideal starting point for a host of neat, self-referential hacks. FORGET teaches SuperBasic to ignore certain keywords. For years I've used FORGET to resolve Toolkit clashes and test variants without resetting the computer. What's more, FORGET can reduce the size of compiled Basic tasks.

WHY lists all the GOSUB lines and procedure calls that have taken place since the last PROC/FN cleared message. The listing appears this month, for those addicted to typing. POP is a devious routine that simulates a RETurn, renaming LOCals and parameters - but *without* returning to the caller. Reader Cyril Doherty suggested this, for use in interpreted error handlers. It's named after a similar feature in SAM Coupe Basic.

Volume C - CHANNELS

This volume combines the Channel functions from May 1988, and USE - the useful default-changing command Sinclair left out of SuperBasic at the last moment. USE lets you type PRINT, INPUT and other channel commands with no need for #4 or whatever after each command and function. Just USE #4 to change the default channel to 4, and USE, or USE #1, to restore the usual default later. The functions come with a dozen small demos that show how you can interrogate QDOS to find the details of SCR, CON and MEM channels, and a discussion of the QDOS channel scheme.

Volume E - ERROR CONTROL

This volume includes PURGE, which quickly removes all tasks but SuperBasic, PICK$ - an alternative to SELECT- plus CHECK%, CHECKF and EDLINE$. Those three functions let you read and validate string and numeric input, avoiding errors that might otherwise stop your program. The volume contains routines and text from January and February 1988, and January 1989.

Volume F - FILE TOOLS

This volume includes 11 files, including GetHEAD and SetHEAD commands from February 1988, and *Customkit* from July. *Customkit* combines small files of extensions into a composite file that loads faster and uses less memory. Thus you can assemble your own 'Custom Toolkit' of favourite commands.

The articles discuss eprom, task and data file headers., The eprom *compiler* lets you give your Toolkit collection a rom header, so that if you program it into a chip your choice of banner appears when you reset the QL, and the commands are linked into SuperBasic from the start.

Volume G - Fast GRAPHICS

The DRAW and PIXEL%, which together form a graphics library much faster and more predictable than standard SuperBasic routines. Four demonstrations are included, plus explanatory text.

The fast graphics routines have been extended to suit the new colours in the Thor's MODE 122; a suitable 16 colour bit-mapped mode is also feasible on the Amiga QDOS emulator, but current versions only use MODE 4.

Volume H - HEAP MEMORY / HOROLOGY

This includes the memory-management keywords RESERVE, DISCARD and LINKUP, from October 1989, plus full documentation and SHOW_HEAP, a utility by Phil Spink which shows the size and location of all areas of heap ram, with the name of the task that owns the memory. SHOW_HEAP can be invaluable if a task crashes and you need access to the data in its buffer. It can also explain where all your free memory has gone, and how to release it. The updated article explains relevant precautions.

```
1000 REMark CHLIST by Phil Spink, tweaks by Simon N Goodwin
1010 REMark v4, expanded to find and list active filespecs
1020 REMark v5, 31/8/89, compatible with MINERVA and THORs
1030 REMark v6, 2.4.90, Turbo Toolkit commands eliminated
1040 REMark v7, 5/8/90, Adapted for QL World DIY TOOLKIT
1050 :
1060 REMark Uses DIY TOOLKIT functions SYSBASE, CHBASE, BPEEK_L
1070 :
1080 DEFine PROCedure CHLIST
1090 LOCal window_flag%,sv_chbas,sv_chtag,ch_num%,address,ch_ty
1100 LOCal sv_chmax,window_offset
1110 OPEN #15,'scr_4x4a2x2'
1120 window_offset=CHAN_ID(#15)
1130 window_offset=window_offset-INT(window_offset/65536)*65536
1140 sv_chbas =  PEEK_L(SYSBASE+120)
1150 window_offset=sv_chbas+4*window_offset
1160 window_offset=CHBASE(#15)-PEEK_L(window_offset)
1170 CLOSE #15
1180 sv_chtag =  PEEK_W(SYSBASE+112)
1190 sv_chmax = PEEK_W(SYSBASE+114)
1200 PRINT \"Channels"\"Number Tag";
1210 PRINT TO 12; "Owner" TO  28; "address" TO 36;"Name"
1220 ch_num% = 0
1230 FOR chtp = sv_chbas TO (sv_chbas+sv_chmax*4) STEP 4
1240   IF PEEK(chtp) <> 255 THEN
1250     address =  PEEK_L(chtp)
1260     PRINT \ch_num% TO 6; PEEK_W(address+16) TO 12;
1270     PRINT JOB_NAME$(PEEK_W(address+10),PEEK_W(address+8));
1280     PRINT " ";TO 28; address TO 36 ;
1290     window_flag% = 0
1300     ch_ty = CHANNEL_TYPE(address)
1310     SELect ON ch_ty
1320     =1
1330       def_slot=PEEK(address+29)
1340       def_block=PEEK_L(SYSBASE+256+def_slot*4)
1350       linkage=PEEK_L(def_block+16)
1360       PRINT MPEEK$(linkage+38,PEEK_W(linkage+36));
1370       PRINT PEEK(def_block+20);"_";
1380       PRINT MPEEK$(address+52,PEEK_W(address+50));" : ";
1390       IF PEEK(address+28)=1:PRINT"In"; :ELSE PRINT "Trans";
1400       PRINT "put file";
1410     =2 : name_at = PEEK_L(PEEK_L(address+4)+8)
1420       name_in$ = MPEEK$(name_at,100):len_name=LEN(name_in$)
1430       x = "'" INSTR name_in$
1440       y = "'" INSTR name_in$(x+1 TO len_name)
1450       z = "'" INSTR name_in$(x+y+1 TO len_name)
1460       IF x>0 AND y = 2 AND z = 2 THEN
1470         name_at = name_at+x+4+1
1480         dev$ = MPEEK$(name_at+2,PEEK_W(name_at))
1490         IF dev$ = "CON" OR dev$="SCR" THEN
1500           IF PEEK_L(address)<128 THEN dev$="SCR"
1510             PRINT dev$;
1520             address=address+window_offset
1530             PRINT "_";PEEK_W(address+28);"x";PEEK_W(address+30);
1540             PRINT "a";PEEK_W(address+24);"x";PEEK_W(address+26);
1550             IF dev$ = "CON" THEN
1560               PRINT "_";PEEK_L(address+108)-(address+120);
1570             END IF
1580         ELSE
1590           PRINT dev$;
1600         END IF
1610       END IF
1620     =0 : PRINT "** Not Known";
1630     END SELect
1640   ch_num% = ch_num%+1
1650   END IF
1660 END FOR chtp
1670 PRINT
1680 END DEFine CHLIST
1690 :
1700 DEFine FuNction IN_LIST(cur_link,link)
1710 REMark Find 'link' in linked list
1720 :
1730 IF link = cur_link THEN
1740   REMark Found
1750   RETurn 1
1760 ELSE
1770   cur_link = PEEK_L(cur_link)
1780   IF cur_link = 0 THEN
1790     REMark End of list - Link not found
1800     RETurn 0
1810   ELSE
1820     REMark Try next link
1830     RETurn IN_LIST(cur_link,link)
1840   END IF
1850 END IF
1860 END DEFine IN_LIST
1870 :
1880 DEFine FuNction CHANNEL_TYPE(ch_address)
1890 link = PEEK_L(ch_address+4)
1900 sv_drlst = PEEK_L(SYSBASE+68)
1910 sv_ddlst = PEEK_L(SYSBASE+72)
1920 IF IN_LIST(sv_drlst,link) THEN
1930   REMark Simple device
1940  RETurn 2
1950 ELSE
1960  RETurn IN_LIST(sv_ddlst,link)
1970  REMark Directory device=1, unknown=0
1980 END IF
1990 END DEFine CHANNEL_TYPE
```

```
2000 :
2010 DEFine FuNction JOB_NAME$(num,tag)
2020 IF  num = 0 THEN
2030   RETurn "SuperBASIC"
2040 ELSE
2050   sv_chbas = PEEK_L(SYSBASE+104)
2060   jb_ptr = sv_chbas+(num*4)
2070   REMark Is this job number in use ?
2080   IF PEEK(jb_ptr) <> 255 THEN
2090     jb_start = PEEK_L(jb_ptr)
2100     REMark Is the tag correct ?
2110     IF tag = PEEK_W(jb_start+16) THEN
2120       REMark Is that an Ident Word (19195 = $4AFB) ?
2130       IF PEEK_W(jb_start+110) = 19195 THEN
2140         RETurn MPEEK$(jb_start+114,PEEK_W(jb_start+112)
2150       ELSE
2160         RETurn "No Name"
2170       END IF
2180     END IF
2190   END IF
2200   RETurn "** Not Valid"
2210 END IF
2220 END DEFine JOB_NAME$
2230 :
2240 DEFine FuNction CHAN_ID(num%)
2250 RETurn BPEEK_L(BPEEK_L(48)+num%*40)
2260 END DEFine CHAN_ID
2270 :
2280 DEFine FuNction MPEEK$(addr,len%)
2290 LOCal p,t$
2300 t$="" : REMark Use Turbo TK PEEK$ for more speed
2310 FOR p=addr TO addr+len%-1
2320   t$=t$ & CHR$(PEEK(p))
2330 END FOR p
2340 RETurn t$
2350 END DEFine MPEEK$
```

Horology is the Science of measuring time. Five commands and one function from May 1989 control four independent stop-watches which tick 50 or 60 times a second. Any task can read, stop or start any of the watches. The instructions explain how to re-assemble the code to provide more timers, if four is not enough.

Volume J - JOBS & Multi-tasking

There are lots of multi-tasking aids around, but the DIY Toolkit ones from April and October 1989 allow full multi-tasking, use very little ram, and are exceptionally compatible.

Peter Postl's *Psion Patch* is a small program that modifies Psion's version 2 tasks to display a cursor, so they multi-task happily with SuperBasic. The patch leaves the task size unchanged, but means you can load with EXEC instead of EXEC W, and use SuperBasic later, or load other tasks.

Peter was inspired by *Taskforce*, Phil Spink's routine that allows switching of input between bad-tempered tasks, and stops greedy tasks swallowing all the memory. Even a 128K QL can edit a 1200 word *Quill* document in memory, alongside 7.5K free to SuperBasic at the same time. *Taskforce* makes an ideal BOOT routine.

Extras include updated documentation, job control commands, and more of Phil's task utilities. JOBNAME TASK displays the name of the newly-selected task every time you press Control C to change input channels. Others let you add names to Psion's tasks, or edit job names in files compiled with Turbo or Supercharge I.17+.

Volume M - MultiBasic and beyond

MultiBasic lets you keep several tokenised SuperBasic programs in one machine, and swap between them instantly under command or program control. Versions 3.2 and 3.3 appeared in March and August 1990. The new MultiBasic 4 can save and restore the display as you swap between SuperBasic tasks. The only limit on the number of screens is available memory, and swapping is very fast. Source and code for versions 3.3 and 4 are supplied.

MultiBasic 4 allocates display space dynamically in the task area, so it does not fragment heap memory. It works wiith Thor Basic, Minerva, *QRAM, Taskforce, Taskmaster* and their ilk. Old windowing environments do not expect SuperBasic to swap, so they don't exchange MultiBasic screens automatically. MultiBasic 4 makes them *more* consistent than before, because screens can swap with all types of task.

Volume N - NETWORKING

This contains material from three months of DIY Toolkit, and more besides; *this large module is only available on disk.* NETPAL appeared in July 1990. It lets you type commands on other computers, linked via the network. You may control several machines from one keyboard. NETPAL includes improved QL, Thor and Minerva programs, plus toolkit commands

The MEM device, updated from July and August 1989, is a complete device driver which lets tasks treat memory like a file. MEM is a flexible way to store or pass information, as it can allocate and share buffers between tasks. MEM works on any QL, but NETPAL expects at least two computers with FSERVE, the remote file server.

Volume P - PIPES and PARAMETERS

This volume includes nine keywords that first appeared in October and December 1988. They let you use 'pipes' in SuperBasic programs, and test the parameters of SuperBasic routines with the flexibility of machine code. Demonstrations include SEARCH_PROG, a routine to locate specified text in the currently loaded program, CAT procedures to show directories in neat columns, and a selective file purger.

Volume Q - QUEUES and TABLES

This volume includes the functions QUEUE%, SYSBASE and CHBASE, from December 1989, with documentation and a dozen SuperBasic examples, including Phil Spink's QLIST, and CHLIST, printed alongside.

CHLIST scans the system and shows the device and file-names, open-type, parameters and other details of all channels currently open on the QL or Thor. It is compatible with *Argos,* Sinclair and Minerva QDOS. QLIST is similar, but shows the input queues that can be selected with Control C. QUEUE% enters characters into a queue as if they were typed at the keyboard, so QLIST helps you get them into the right channel.

Volume R – REPLACE and utility functions

REPLACE was born in the June 1988 *QL World.* It consistently changes variable, loop and device names throughout a SuperBasic program. Use it for bulk editing, to resolve name clashes, or to make programs more readable. REPLACE is very fast and context-sensitive; it does not change things it should leave alone, like text in quotes and strings. Volume R also includes the utility functions NEWCHAN%, LOCKUP%, LOWER$ and UPPER$, which I introduced last month.

Old Masters

These twelve volumes expand on the material published from January 1988 to date. They do not include the first DIY Toolkit articles, written by Marcus Jeffrey in 1987. We may collate early DIYs if the current volumes are popular and readers ask after earlier ones like TUNE, SPEED, LINE3D, POKE_$ PEEK_$, G_SAVE and G_LOAD, if Marcus agrees and can find the original files. This will take a while in any case; the routines need testing on later roms and some bugs need correction.

Overview

The revised DOC files supplied with the bundles total over 50,000 words. They have been updated to match code improvements, document the extra demos and utilities, and to reduce the need for cross-referencing between files.

The revisions are expanded from my original text, whereas all the words printed in the magazine are re-typed by outside type-setters, who were rather erratic in the early days.

The revisions include jokes edited out by our erstwhile sub. Commas, underscores, zeroes, and spaces take their

intended places, and 'umlaut' no longer appears as 'amulet'.

You can print or edit these _DOCs with *Quill* or *Xchange*. A printed introduction comes with the disk or cartridge, along with photocopies of freehand diagrams that do not suit *Quill*; again, these come straight from my original copy.

The _CODE and _ASM files include 50-odd resident procedures and functions, assembled with HiSoft's DevPac. I have checked and tweaked all the _ASM files to suit Metacomco's Assembler, which is fussy about 'generic' instructions which DevPac and Sinclair-endorsed assemblers swallow without complaint. Each CODE file has a matching SuperBasic _BOOT file to load it. Line numbers differ for each, so _BOOT files can be merged.

SuperBasic examples have a three-part name, starting with the volume name, followed by a description of the example, and the extension _BAS. Be sure to load the appropriate Toolkit keywords before loading these programs, or you may see a 'BAD NAME' or 'ERROR IN EXPRESSION' report. If so LRUN the appropriate _BOOT file to load the extensions, then re-LOAD the SuperBasic.

You can freely use the binary CODE files, or re-assembled variations of them, in complete programs which may be sold or given away, but you may not charge anyone for the DIY Toolkit machine code. _BAS, _ASM and _DOC files are copyright. Each purchaser is licensed to edit or modify them, but may not supply them, or the information therein, to others without written permission from the authors.

Ordering

Each volume costs £3. Future toolkits will be bundled in a similar way, available on the same basis as the corresponding issue of *QL World* appears.

There is a processing charge of £4 per order. Thereafter, each bundle costs £3, so the total price of bundles goes down as you order more. The minimum charge is thus £7, for one volume, but you can buy two volumes for £10, three for £13 (£4 + £3 x 3), and so on. This formula reflects our wish to make DIY Toolkit routines available to as many readers as possible, and the full cost of processing small orders.

Please make payment to DIY TOOLKIT, in British Pounds, Sterling postal orders, drafts or Eurocheques; all cheques drawn on UK banks are acceptable; CGH cannot accept Credit Cards, foreign funds, ECUs or IOUs.

We are taking a calculated risk in choosing these low prices. We offset this by splitting the Toolkit into sensible lumps, so that you don't have to pay for more than you want, and can come back for more later, if you like your first helping.

We do not use second-hand cartridges or 'unbranded' disks. They may be fine for use at home, but here they would represent false economy. We want to minimise the inconvenience and delay of sending replacements. DIY Toolkit uses branded and guaranteed Sony disks, and new microdrive cartridges from this year's production at Ablex. We will promptly replace media damaged in transit or return of the faulty item.

CGH duplicates media to order, to ensure that everyone gets the latest version. Disk duplication get priority, as disks can be copied and packed more easily than cartridges, but CGH Services are established and well-organised; I am confident that they will process all orders efficiently. **Please make it quite clear which volumes you want.**

Technical enquiries and suggestions will be forwarded to the authors. I have done my best to anticipate questions in the revised documents. Please be patient if you weant personal advice; I'm keen, but there are lots of you.

Delivery will be as fast as is practical; CGH uses First Class post in the UK, and Air Mail to Europe. Please indicate your preferences if you live further away, or send an extra £2 for *Swiftair* service. "*Swiftair* provides accelerated treatment in the UK and express delivery abroad where available", according to the Post Office

Write to DIY Toolkit (Orders), Cwm Gwen Hall, Pencader, Dyfed, Cymru SA39 9HA, or call Richard Alexander at 0559 384 574.

text87 version 3.00

offers today's state-of-the-art user-friendly environment for document production. With integrated spelling-checker and extensive support for highest quality printing possible on daisywheel and 9-pin printers.

fountext88 and **founted89**, the graphic printer-driver and editor for text87 provide graphic founts without the limitations in text editing and document size imposed by QL desktop-publishing programs. More than 35 quality founts of up to 72 pixels in height are supplied and new founts can be designed or captured from saved screen images.

2488, the state-of-art text-mode printer drivers for selected Epson LQ, NEC P, Star LC and Panasonic KXP 24-pin printers offer advanced features such as multiple typefaces, proportional spacing, micro-justification, double height, shadow and outline modes (depending on the printer model).

New Releases

text87*budget*, has all the features of text87 version 3.00, the only limitations is that it cannot be used with 24-pin and laser printer drivers. Upgrade to the full version will be availabe at the price difference.

typeset90-deskjet text-mode printer drivers for Hewlett-Packard Deskjet and Deskjet plus support printer's internal founts plus the full range of Roman and Helvetica plug-in cartridges.

typeset90-laserjet text-mode printer driver for Hewlett-Packard Laserjet series printers supports printer's internal founts plus a wide range of fount cartridges (incl. Roman, Helvetica, Universe) in sizes between 6 points and 72 points.

typeset90-GQ5000 text-mode printer drivers for the Epson GQ-5000 supports printer's internal scalable and bit-mapped founts in sizes between 6 points and 72 points.

text87budget: £45 ● text87: £60 ● fountext88 + founted89: £40 ● 2488: £15
typeset90-deskjet: £20 ● typeset90-laserjet: £40 ● typeset90-GQ5000: £40
special offer extended to 15 October: £10 discount on orders of £100 or more

Software is available in English, French, German and Italian. Prices are inclusive of airmail. Payable by cheque, Postal Order or Eurocheque. Please specify language and disk system (all 31/2" and 51/4" formats can be supplied). text87 requires at least 256K memory expansion.

Software87, 33 Savernake Road, London NW3 2JU

ARCHIVE DATES

Archive may not be able to change history, but it can rearrange dates to suit itself, says Mike Lloyd

Many QL owners are apt to forget that two programming languages came with their computer: *SuperBasic* and *Archive*, the programming language for the database bundled in the Psion package. Psion are no longer developing Archive and more recent database software on the MS-DOS market tends to outperform it, but Archive is a powerful and attractive language despite its limitations. Using it seriously is often a labour of love, but the results are usually worthwhile and occasionally astonishingly good.

One of Archive's more serious omissions is the lack of a comprehensive data management suite. This is all the more remarkable bearing in mmind that most databases contain date fields of some sort or other and that data is very often manipulated or searched for according to a date value. Accompanying this article are a set of a useful Archive procedures which fill the gap left by the original authors of the language.

Archive has only numeric and string field types for its database records whereas most of its rivals have specific date fields with special attributes. Dates must therefore be held in Archive as either a number or a string and be manipulated either by specially written procedures or by the three date-related functions included in the language.

The first date function in Archive is DATE which converts today's date into one of three date display strings according to a numeric parameter, as follows:

Value	Date String
0	"YYYY/MM/DD"
1	"DD/MM/YYYY"
2	"MM/DD/YYYY"

The first string puts the most significant value first, presumably in order to simplify date comparisons. The second value is a traditional British format and the final option is the muddled format favoured by Americans and Japanese digital wristwatches.

For my money, none of them is altogether suitable for displaying dates. They are all numeric, leading zerios must be used for values less than ten, and the use of the slash creates a solid-looking body of characters which is difficult to read quickly and accurately. My preference is for the month to be represented by three letters and the year by two digits with all the elements separated by spaces, such as:

12 Mar 90

By passing a date in Archive's "Type 0" date format to the function DAYS it is possible to obtain the number of days which have elapsed since the start of Archive's calendar on 1 Jan 1583, the first complete year in the Gregorian calendar. Known as a relative date, this method of storing date values is commonplace across all types of databases. it allows ready comparision between dates and it is possible to carry out date arithmetic, such as finding the date sixty days after today's date. Archive reduces the value of relative dates, however, by not including a function to convert them back to date strings.

The final date-related function is MONTH, which converts a number into the name of the appropriate month of the year with 1 producing "January", 2 producing "February" and so on. The modulus of numbers greater than 12 are used so that 13 again produces "January".

In the Archive applications which I produce commercially I have standardised all date displays to the "18 Jan 90" type of format. Because dates stored in the database are displayed far more often than they are used for comparisons all of my date fields hold the date in the form in which they are printed on the screen. However, in order to perform date arithmetic, for instance to search for records created after a given date, a procedure is required to convert date strings to their relative date equivalents. Occasionally it is necessary to convert in the opposite direction so that relative dates become date strings. The first two listings show procedures which carry out these conversions.

Listing 1 is responsible for turning relative dates into text strings, hence the

LISTING 1

```
proc VALtoTEXT;tempday
  local feblen
  let DAY=tempday-145001
  if DAY<0: return : endif :rem 0 = 31 Dec 79
  let YEAR=int(DAY/365.25)
  let DAY=DAY-int(YEAR*365.25):rem corrects Leap Year
  if (1980+YEAR)/4=int((1980+YEAR)/4)
    let feblen=29: else : let feblen=28
  endif
  if DAY<=31: let MONTH=1
    else : if DAY<=31+feblen: let MONTH=2: let DAY=DAY-31
      else : let DAY=DAY-31-feblen
      if DAY<=31: let MONTH=3
        else : if DAY<=61: let MONTH=4: let DAY=DAY-31
          else : if DAY<=92: let MONTH=5: let DAY=DAY-61
            else : if DAY<=122: let MONTH=6: let DAY=DAY-92
              else : if DAY<=153: let MONTH=7: let DAY=DAY-122
                else : if DAY<=184: let MONTH=8: let DAY=DAY-153
                  else : if DAY<=214: let MONTH=9: let DAY=DAY-184
                    else : if DAY<=245: let MONTH=10: let DAY=DAY-214
                      else : if DAY<=275: let MONTH=11: let DAY=DAY-245
                        else : let MONTH=12: let DAY=DAY-275
                      endif
                    endif
                  endif
                endif
              endif
            endif
          endif
        endif
      endif
    endif
  endif
  let DATE$=num(DAY,2)+" "+month(MONTH)(1 to 3)+" "+str(80+YEAR,2,0)
```

name VALtoTEXT. The single parameter passed ot it is treated as a relative date in the Archive calendar. To avoid complexities caused by the rule that only century dates divisible by 400 are leap years (so that 1900 was not a leap year, but the year 2000 will be) a new relative start date was chosen. By happy coincidence, 31 Dec 79 is Day 145000 in Archive's calendar and so it was used as the earliest valid date. This has proved to be entirely suitable for databases dealing with modern events, but QL owners using Archive for historical purposes will need to adapt these procedures slightly to work with earlier dates.

Leap years

Leap years are a problem because of the extra day in February, and so a special local variable is declared to represent the length of February. Global variables are used to represent the day, month and year (in two digits). All global variables are identified throughout the listings by being shown in capitals. After setting DAY to its new relative value and testing to see if it is still positive the year is calculated by dividing DAY by 365.25, the correct number of days per solar year. DAY is then reduced by the number of days in the whole years and the length of February is calculated according to whether the current year is a leap year or not.

It is now necessary to calculate which month the relative date is in. Unfortunately there is no pattern to the number of days in each month and so twelve nested IF statements must be worked through in order to arrive at the correct month. The principle is simple: after taking away the whole number of years, if the remaining days number less than 31 the date must be within January, if the remainder is more than 31 but less than 59 then the date must be in February, and so on. Because of February's vagaries dates from 1 March onwards are worked out using 1 March as the base date.

Whatever month is calculated a remainder will be left, representing the day of the month. At the end of the nested IF statements, therefore, the variables YEAR, MONTH and DAY are all set to values which can be used to create a date string. Archive might not be blessed with many date-related functions but it is positively overflowing with functions for converting numbers into text and vice versa. The NUM function is used to convert the DAY variable to a string so that there is a leading space for dates earlier than the tenth of each month. This makes it easier to align dates in columns.

The MONTH function returns the entire name of the month and so just the first three characters are selected. Incidentally, unlike SuperBasic's fussy treatment of reserved keywords, Archive allows words to be used as variables even though they are keywords. The "month (MONTH)" clause causes no problems at all.

The year value is converted with the STR function because alignment is not an issue. The date elements are linked together by spaces and held in the global variabel DATE$ which retains its value until the next call to the VALtoTEXT function.

The reverse process is much easier because the Archive DAYS function can be used to convert date strings into relative dates, provided that the string is in the right format. **Listing 2** shows the TEXTtoVAL procedure which changes dates like "9 Oct 92" into the required Type 0 format ("1992/10/09") before producing a numeric relative date value.

Three local variables are declared to hold the day, month and year values. D$ is found by slicing the first two characters from the date string passed to the procedure. If the leading character is blank it is replaced by a zero. The month, of course, is a three-letter abbreviation. The numeric equivalent is found using the INSTR function which works in a similar way to the SuperBasic equivalent, albeit with a more conventional syntax. The number is converted to a string in the same multi-function command. Again, if a leading zero is required it is added to the front of the string.

All of the date elements can now be assembled into the correct order, not forgetting to give the year a four-figure format. Because Archive differs from SuperBasic by passing parameters by value rather than by reference it is possible to treat Archive parameters as local variables. The tempdate$ variable is therefore put to good use as the final destination of the date elements before the days function produces a relative date value. As with the DATE$ variable in the previous procedure the result is put into a global variable, this time called DATEVAL, which can be accessed outside the TEXTtoVAL procedure.

Applications

Having achieved the essential conversions from text dates to relative dates and back again it is possible to use the procedures within others in order to make full use of dates in database applications. Three examples are used to illustrate this article. The first finds the date of the Friday immediately following a given date, of use in applications where "week ending" dates are important. The second calculates the period between two dates in days, weeks, months and years. The third procedure takes a date and a period of days as its argument and produces the date at which the period ends. This allows a database to display the date of an invoice and the date on which the thirty days' grace will expire, for example.

As explained earlier, I habitually hold dates as strings to accelerate screen refreshes and so my procedures tend to take date parameters in the "18 Jun 91" style.

LISTING 2

```
proc TEXTtoVAL;tempdate$
  local d$,m$,y$
  let d$=tempdate$(1 to 2)
  if d$(1)=" ": let d$="0"+d$(2): endif
  let m$=tempdate$(4 to 6)
  let m$=str((instr("JanFebMarAprMayJunJulAugSepOctNovDec",m$)+2)/3,2,0)
  if len(m$)=1: let m$="0"+m$: endif
  let y$=tempdate$(8 to 9)
  let tempdate$="19"+y$+"/"+m$+"/"+d$
  let DATEVAL=days(tempdate$)
  endproc
```

LISTING 3

```
proc WeekEnding;tempdate$
  local tempdate
  TEXTtoVAL;tempdate$
  let FRIDATE=int(DATEVAL/7)*7+6
  VALtoTEXT;FRIDATE: let FRIDATE$=DATE$
  endproc
```

LISTING 4

```
proc DateDiff;date1$,date2$
  local date1,date2
  TEXTtoVAL;date1$: let date1=DATEVAL
  TEXTtoVAL;date2$: let date2=DATEVAL
  let DAYGAP=date2-date1
  let WEEKGAP=int(DAYGAP/7+0.5)
  let MONTHGAP=int(DAYGAP/30.44+0.5)
  let YEARGAP=int(DAYGAP/365.25+0.5)
  endproc
```

LISTING 5

```
proc EndDate;tempdate$,period
  TEXTtoVAL;tempdate$
  let EDATE=DATEVAL+period
  VALtoTEXT;EDATE
  let EDATE$=DATE$
  endproc
```

Some changes to the following listings will be needed if you choose to adopt another standard for displaying or storing dates.

The WeekEnding procedure takes a single date as a parameter, which it converts to a relative date using the TEXTtoVAL procedure described above. The relative date is divided by seven, the remainder thrown away, and then multiplied by seven to give the relative date of the Saturday preceding the day in question. By simply adding six to this value the relative date of the Friday *following* the given date is found. This is converted into a string using the VALtoTEXT procedure. Note that at end end of the process there are two useful values available: FRIDATE, the relative date, and FRIDATE$, its text equivalent.

Relativity

DateDiff takes as parameters two dates, the second one assumed to be later than the first. Both are converted to relative dates with the resulting values held in local variables. The difference in days is quickly found by subtraction and stored in the DAYGAP variable. The difference interms of weeks, months and years is calculated by dividing this figure by a constant, but it can be seen that some approximation is accepted in order to keep the procedure simple. The addition of ·5 to every value before its integer is found ensures that values are rounded to the nearest whole number.

The final procedure follows the pattern set by its predecessors. It takes a date in text format and a number of days as its parameters. The relative date equivalent of the date string is calculated and the period value added to it to form EDATE, the relative date of the end of the period. This variable is then passed to the VALtoTEXT procedure to obtain its text equivalent which is stored in the variable EDATE$. These two values can then be used outside the procedure until it is next called.

To tie all of the previous listing together a demonstration procedure has been included (and called "start") which can also be used as a testbed for the date utilities.

LISTING 6

```
proc start
  cls : let col=50
  print "The QL Clock reads:-"; tab col;date(1)
  if days(date(0))<=3
    print "Adjust to today's date and try again": stop
    endif
  print "Within your application the date is:- "; tab col;
  VALtoTEXT;days(date(0)): let TODAY$=DATE$: print TODAY$
  let YR1990$="01 Jan 90"
  print YR1990$;" can be converted to day number:- "; tab col;
  TEXTtoVAL;YR1990$: let YR90=DATEVAL: print DATEVAL
  print "... and converted back again to:-"; tab col;
  VALtoTEXT;YR90: print DATE$
  print "The first Friday in the year was:-"; tab col;
  WeekEnding;YR1990$: print FRIDATE$: let FirstFri$=FRIDATE$
  print "Next Friday's date is:-"; tab col;
  weekending;TODAY$: print FRIDATE$
  print "The difference between the 2 dates is:-"
  DateDiff;FirstFri$,FRIDATE$
  print tab 10;DAYGAP;" days, or ";WEEKGAP;" weeks, or
  ";MONTHGAP;" months."
  print "In ten days time it will be:-"; tab col;
  EndDate;TODAY$,10: print EDATE$
  endproc
```


SPECIAL OFFERS ON KEYBOARDS !!

Standard QL replacement keyboard, with black enclosure	£ 22.00
Anti-bounce device for standard keyboard	£ 5.50
KBL 128 all-purpose computer enclosure	£ 28.00
64 way ‹Flip back› connector for KBL 128	£ 8.00

All the above items come with full instructions.

All prices inclusive of V.A.T.

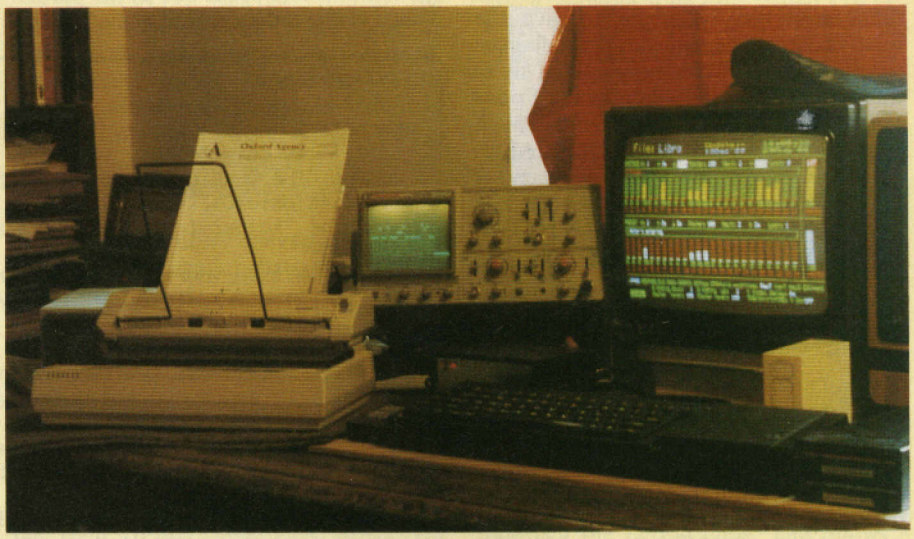
P.& P. free of charge in U.K. & N. Ireland. Phone or write for details of cost for shipments abroad.

Send your order, with cheque or postal order, to:

KEYBOARD PRODUCTS LTD.
UNIT E2, WELLAND IND. ESTATE
VALLEY WAY
MARKET HARBOROUGH
LEICS. LE16 7PS
Tel. 0858 410048 Fax. 0858 66786

ONE MAN'S

Edwin Osborn tells how he started with Quill and developed his QL dramatically.



My system: the oscilloscope is displaying output for the lighting dimmer racks, the small box behind and to the left of the QL is the dimmer-board interface, the fireman's cap is used to protect against morning sun through the windows!

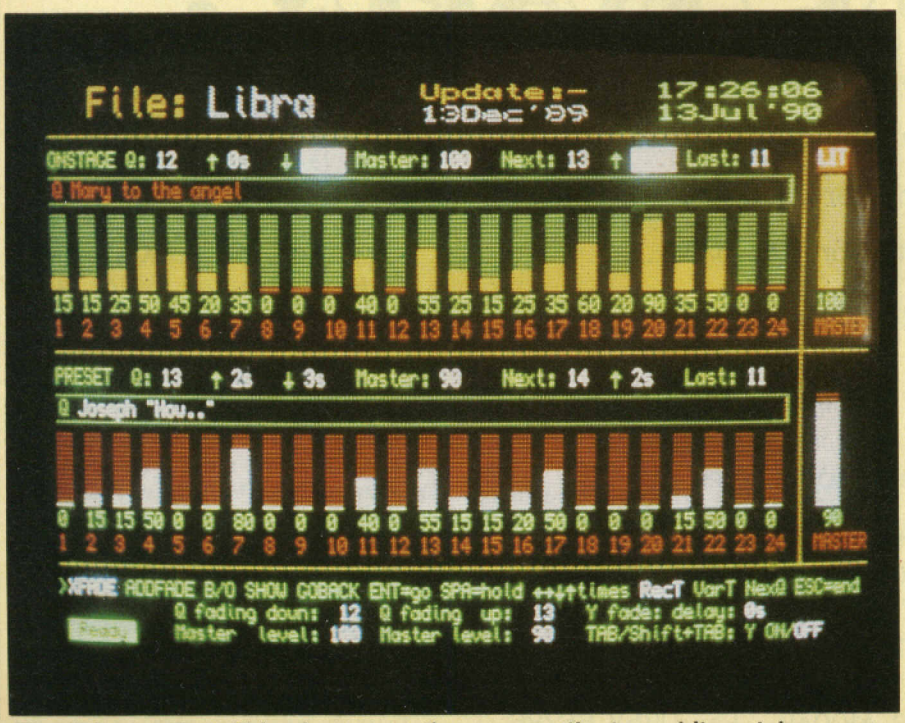
My first real first-hand experience of the electronic processing age was the purchase of a Brother EP44 typewriter/printer about five years ago — which quickly led to my coveting a micro as soon as I could afford one. As a step toward making better use of the EP44, I succeeded in tape-recording and replaying the EP44's memory with the help of a surprisingly simple home-made interface. Sadly, practical application was limited since one could only print the tape-recorded memory — not feed it back into the typewriter's memory for further editing.

At about the same time, the office where I was working purchased a similar-system Canon and turned out to have an otherwise unused BBC micro. So I borrowed the latter as often as possible, both to drive my EP44 and in an attempt to persuade my office to computerise, if only to do a little word-processing. My programs for the BBC (written 'from scratch' in Basic and Wordwise Plus Basic solely with the help of the handbooks) were unfortunately too clumsy to persuade anybody that computers could be seriously useful, but the main problem was that operators almost invariably hit the Break key at some crucial point or failed to follow prompts, and the Wordwise-Plus wordprocessor is certainly not the most user-friendly.

However, the experience was valuable. When I first started programming I spend a nightmare three weeks not realising the computer was faulty. I now always remind myself that a problem might be electronic and not be in my programming. I learnt a lot about using printer code books and a little about the use of procedures. I formed some working opinions about the use of computers, including (i) that a near-static screen display is best for serious use and (ii) that computers ARE a nuisance if not carefully programmed to do just what is needed — preferably for something not easy to do in any other way.

I purchased my first 128K basic QL three summers ago, to use initially with the EP44 as printer and a rented SONY 12 in colour TV as monitor, a surprisingly good screen. I found I could use MODE 4 and CSIZE O,O quite happily.

By now I had seen and tried various wordprocessors and was consequently very pleased with Quill. It seems an almost ideal wordprocessor for the many people who want an instantly usable wordprocessor and who are scared and bewildered by superb but complex alternatives. I now "teach" basic wordprocessing part-time to 17-25 year old students



The "Dimmer Board" in-show operating screen: the two white patches near the top show highlighted fading times in black on white.

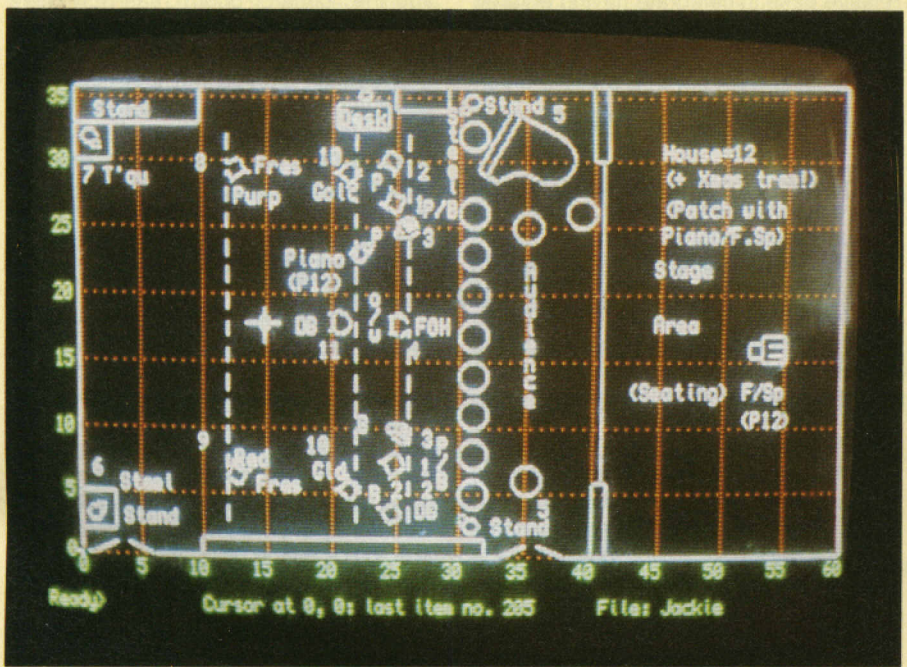
SYSTEM

and they certainly do not find other programs easy to use — nor do staff at the same college which has just computerised its offices.

Needless to say, when it came to tax returns (I am self employed and do my own accounts), Abacus did the job splendidly, and also for a friend for whom it saved £15,000 in possible back-tax when I was able to produce acceptable accounts, mainly thanks to the QL.

At this stage I was also about to start up a small business as a tutorial agency, requiring filing and mail-merge type facilities, so I launched into programming Archive for this, mail-merging clumsily with Quill. Neither system was very satisfactory, and I constantly tried programming my own facilities — and constantly met what became the dreaded "out-of-memory" message edited — and I was occasionally confused by the Quill bug that repeats "ghost" words when re-adjusting line-filling. (I have never worked out the exact sequence that causes this — I merely cure it by deleting one letter of the repeated word or phrase, when Quill re-adjusts and prints the correct text, less the letter just deleted.)

The CUB monitor revolutionised my feelings about computing, with its superbly "still" display (I have not seen another monitor on any machine quite so restful, and I have seen many almost painful to use) but created an alarming problem: plugging the (switched-on) monitor in with the computer on created, first, "strange effects", and then blew the video-driver chip! I made this my

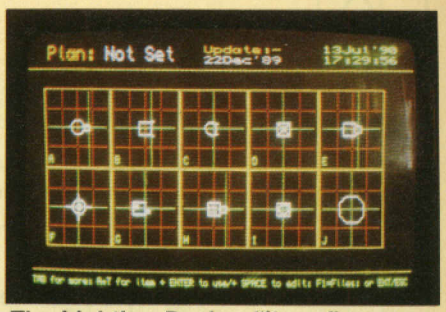


The Lighting Design full-screen display with plan in place: a reduced-window display is available allowing room for the menu and clock.

just when the program began to seem useful!

Then I received a commission to type up large numbers of A/level and GCSE test papers for which I was loaned the 14 in CUB monitor I still use, and used Quill highly successfully for this in combination with a loaned Brother Daisywheel printer. The only snags I met were that Quill sometimes locked up when scrolling upwards past a page break in a document that had been severely excuse to buy a second QL with 640K memory expansion as well as repairing the original.

I received advice that diodes in series with the video lines might cure the problem (caused by static) but this proved unusable. The chip survived but the RGB elements of the picture became displaced from each other, an entertaining but unsatisfactory effect. To this day I am merely careful to plug in (and so switch on) the computer last — a pity in that I



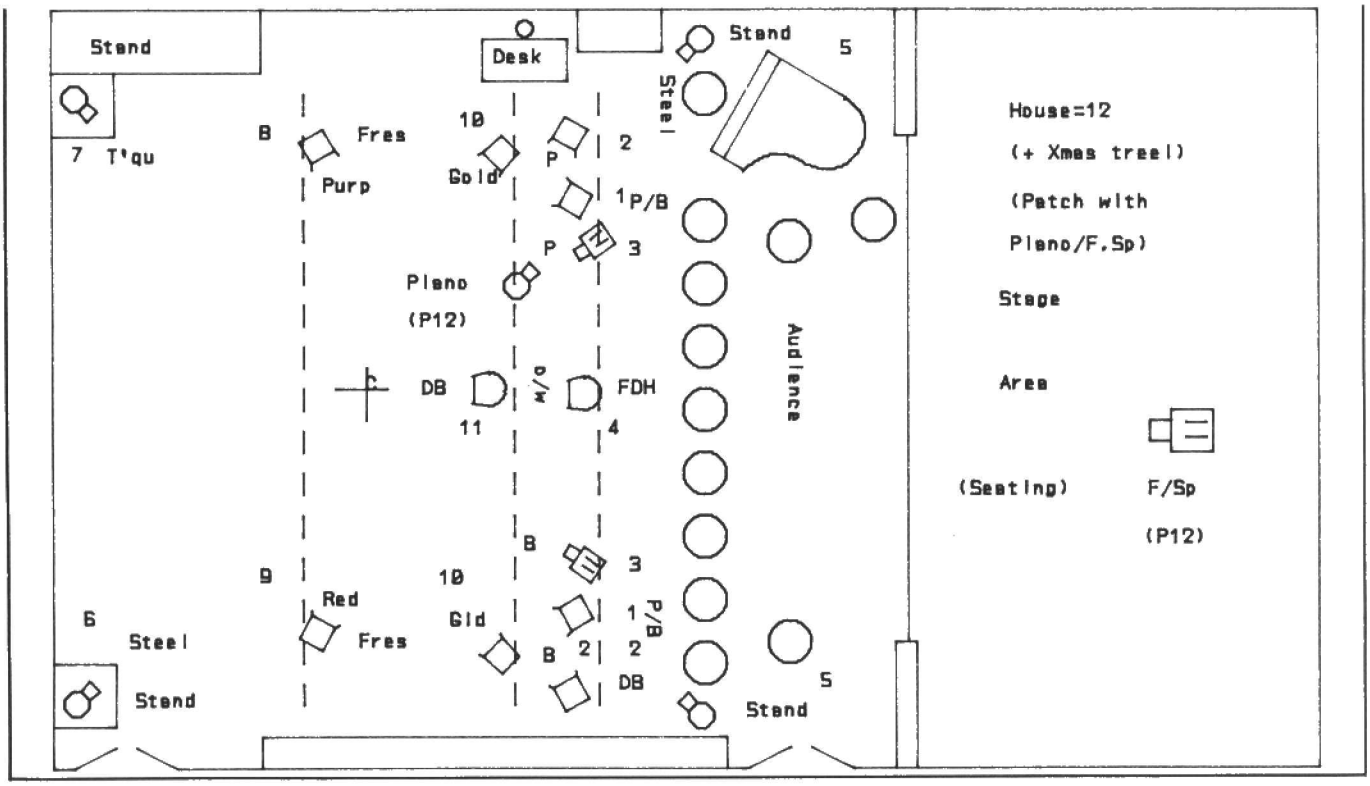
The Lighting Design "items" screen (one of two): editable symbols can be placed on the main plan and then rotated/enlarged/shrunk as required.

used to be able to leave on just the computer (continuously) to maintain the clock.

The typing commission provided funds to purchase a Brother M-1209 dot matrix printer, and I moved on to typing up French, German, Chemistry, Maths and Physics papers with elaborate use of the Quill translate codes allowing me to print almost any symbol. I overcame the Quill translate limitations by a mixture of having several Quill microdrives with different printer data set-ups and typing directly symbols that were incorrect on screen but correct when printed. Not ideal, but not particularly difficult with a little practice and a "translate sheet" stuck in full view. Incidentally, I now use this latter method constantly to produce imperfect but useful tabulation boxes — underline for the horizontals and CTRL+SHIFT+S for the verticals which by chance produces a reasonably suitable symbol on screen: an upside-down "!".

With the extended memory, Quill never locked up again, but the out of memory message alarmingly still bugged my own filing program, to my dismay, until I found a solution. I note this here because it may be of use to others still learning like myself. My explanation may not be entirely correct, I hasten to add, but I believe it is essentially correct.

The handbook encourages one to make use of the fact that one can pull out a single item from a string array simply by giving parameters. In my case, I had a file entry store DIM rec$(200,12,40). To edit file entries, I printed the "active character" with paper 2 (giving a red cursor block), using a line like AT y,x:PRINT rec$(r,l,c). However, it seems that this instruction causes the QL to make a copy of the whole array, from which it then prints the one item — thereby rapidly using up blocks of heap which become inaccessible for use later. The solution was to use a "carrier" string, preferably dimensioned, so that with arrays DIM rec$(200,12,40).c$(1), the printing instruction line becomes: c$=rec$(r,l,c):AT y,x,:PRINT c$. Sud-



denly the QL's memory became (relatively) cavernous.

Around the same time I discovered that although it was possible to make loops by having a procedure call itself (recursive loop), this both appeared to use more memory (I understand the computer makes a copy of the procedure variables each time round) and — more obviously — resulted in a bumpy display. For example, when moving a "cursor" as created above along a line, this would move smoothly for about five places, the "glitch" a moment before moving on. The solution, was, of course, to use normal REPEAT loops.

At this stage, with a 20-page filing program, compilation became essential. I found and find the budget Q-Liberator excellent and essential. I also added DP Lightning — another essential. The filing program started to look and feel like a piece of commerical software, and completely satisfied my criteria for usefulness. In combination with a mildly bugged self-written word-processor/mail-merge program, I could send out fifty and more letters in half a morning and keep legally required reference files of my dealings with clients. Incidentally, my clients' reference numbers are generated entirely by the computer and lead directly to their reference files "sideways" — mdv2 held the main files, which could be cut about and modified at will, while mdv1 held the variable-length reference files simply named and called-up from any main file by the reference numbers.

With compiliation, speed enhancement and more programming experience I began to be more than a little addicted — and increasingly pleased with what the QL could achieve. For some time I had felt like an underdog compared with those able to afford more expensive and more socially-acceptable systems, but now I began to realise that — as many *QL World* articles had included to my (then) surprise — it was true that the QL is in many ways more usable and useful than many more expensive machines. I wrote a version of pocket Halma (or Chinese Checkers) with relative ease to celebrate having entered successfully the Critical Mass game program printed in *QL World*.

Miracle System

A new project now suggested itself — to write a program to make the QL look like a manual theatre-lighting dimmer board (see the screen-dump) and operate like a computerised one, which would operate the lights via a suitable interface to standard dimmer (Strand) units which use a multiplexed input. Further, to write a linked lighting-design plan-drawing program — and both simple enough for students to use without hours of training.

Funds arrived happily in the middle of this project allowing me to upgrade to an 896K Trump card with Miracle systems twin disk-drive — and (of course) my QL's capability took another leap upwards. Screens can be snapped into place by LBYTEing from ram memories: loading and saving is apparently infallible (although I did not have much trouble with microdrives and regard them as underrated — they only failed me as often as I have known over-used BBC/Cumana driven 5.25in. disks to fail, ie, rarely but always when it really matters! Psion printer drivers are "transparent".

Building the interface was an agreeable challenge — I decided to use the QL SER1 output at 9600 Baud on the grounds tht any errors in my electronics would then not blow any chips in the QL. The major problem was to get the QL to feed out data fast enough to keep lighting fades smooth and in general to produce real-time responses where a tenth of a second's unpredictable delay can be too much. Carefully planned loops and use of the KEYROW function produced the required effects (simultaneous keying of more than one option is possible with this) and the "dimmer board" has now been used successfully on three productions. On two of the productions, the progam was actually proved not just useful but essential — I was able to pre-plan "blind" complete sequences of cues which, in the event, I only had time to check briefly before showtime. Other advantages of the program over standard lighting systems are that one can have cue lines on screen and text labels identifying lantern function.

The linked lighting-design planning section of the program is not yet complete, needing its own little wordprocessing section for notes and also help screens, but has already proved very usable. A major criterion here was simplicity of operation — which I felt I must have achieved when my 8-year old neice confidently drew a Humpty Dumpty on a wall on a stage with looped curtains! Another was to be specific to lighting design — the program allows one to check lantern coverage both via "light pools" on the "floor" of the plan and via vertical cross-sections, complete with stick-man.

I found I needed to write my own screen dump to print genuinely scale diagrams printed horizontally or vertically (still 2% out — and taking a while to print). These dumps run as multi-tasking tasks, and I have included the listing for the horizontal dump since others may find it useful, if slow. Just recently, however, I have found further use for the plan-drawing program, producing maps for my archaeologist financée. For this purpose, the dot-matrix print-out quality proved unacceptable, but the program stores graphics data in a format ideal for driving a pen-plotter such

as the Epson HI-80. Using this, good-quality maps and plans can be produced in acceptably short times — about two minutes compared to twenty for an A4-sized (vertical format) dump (the horizontal dump takes about 12 minutes).

I am still working on the aforementioned word-processing 'notes' section — my experience with the real-time needs of the dimmer board program leading to some satisfyingly fast responses from the program compared to my earlier efforts. However, I recently found that I still had not entirely resolved the memory use question — statistics from the Liberator still revealed alarming heap usage when entering text.

It appears that what was happening was this: when inserting a character into a string, eg c$ into 1$, originally dimensioned with DIM c$(1),1$(40), this can be done with a line like:

1$=1$(1 to 5)&c$&1$(6 to 40).

(Note: I realise this loses 1$(40), — the line is merely an example to make the point.)

Two things, however, appear to be true: it is essential that 1$ was originally dimensioned if the text in 1$ might be less than 40 characters and it stops memory wastage to dimension two strings and swap them around. With, for example, DIM1$(40),la$(40),c$(1) the line above becomes:

la$=1$(1 to 5)&c$&1$(6 to 40):1$=la$

It may not, of course, be necessary to switch back to 1$ immediately — la$ may become subject to another process, and the modified string be handed back to 1$ at that stage.) With all string handling done in a similar way, it appears that heap usage never increases at all once the DIMing has been performed.

Nuisance

A further programming point that may be of interest is that I now always use a procedure to completely trap all keypresses during keyboard input, ie use INKEY$(-1) to read each keypress even when, for example, calling for a filename. This avoids any possibility of trying to coerce a string and one can allow only specific inputs quite easily, for example, only one decimal point or only a minus sign at the start of a number. Further, the input can always respond to special keypresses such as F1 for help or ESC to abort — or it is easy to offer a name or number which vanishes if any acceptable-entry key is pressed. The "nuisance" is that deletion of errors must also be programmed — the editing facility of the INPUT statement is not, of course, available: I usually allow only "delete left" from the end with CTRL <-, or just <-, and do not allow cursor movement along the entry.

As will be evident, I program entirely in SuperBasic and rely on the Q Liberator to produce machine code. I have tried a little Assembly code programming, and read much about it, but my objectives always seem to turn out to be some enormous project, and I find the thought of working entirely in Assembler too daunting! I justify myself by arguing that one point of having a computer is that it works out the "boring coding" for you from an understandable language.

I hope this account may be of interest and encouragement to other QL readers. From a complete novice it is possible to progress to complex programs that are really useful without too much strain. Apart from the original handbook, I found useful ideas in Donald Alcock's *Illustrated Super Basic on the QL* and, absolutely essential, in various articles in *QL World* — otherwise entirely teaching myself. I must admit a little pride in the range over which I now use my QL — wordprocessing, filing, mail-merging, accounts, to operating and planning theatre lights and, now, drawing very usable maps — all serious jobs with very real output. The recent purchase of a cut-sheet feeder has even left me free to take lunch (or whatever) while the computer churns out my business letters!

I am now developing my next project — programming for intelligent control of a model railway, and the electronics for reading data both in and out via Ser1 is functional. I have always had this use of a computer/electronics as a lurking ambition since I first used transistors to switch signals 26 years ago, when, as a teenager, it was, yes, Clive Sinclair handbooks that led me into the mysteries of making transistors do things!

NEWSAGENT ORDER FORM

By placing a regular order with your newsagent you can be sure of receiving *Sinclair QL World* every month. Complete this coupon and pass it to your newsagent.

Dear NEWSAGENT,

Please reserve/home deliver _____ copies of *Sinclair QL World* every month for:



Mr/Mrs/Miss .....

Address .....

.....

Telephone No.:.....

Sinclair QL World is published on the last Thursday of every month.

Any distribution queries to: **SM Distribution Ltd, 6 Leigham Court Road, Streatham, London, SW16 2PG. Tel: 081-677 8111.**

INTERNAL NUMBERS ON THE QL

Integers and real numbers are two ways of handling numerical values. Simon Wallis explains how the QL uses them.

This article shows how the QL represents numbers internally, explores the limitations of the Integer and Real datatypes, and implements arithmetic operations with unlimited precision.

There are two numeric datatypes available in SuperBasic. Programmers who have progressed from more humble micros to the QL may be familiar only with Real numbers; there are occasions when Integers are useful, too. An Integer, as far as SuperBasic is concerned, is a *whole* number between −32768 and +32767. These limits are chosen because they happen to be the largest numbers that the QL 68008 processor can happily deal with in one go. Logically, Integer operations should be a lot quicker than their Real counterparts, but SuperBasic is an interpreted language, so the QL spends most of the time when RUNning a program in trying to undertand what you want it to do, rather than actually doing it. If you possess a SuperBasic compiler, you will probably notice that the integer arithmetic is several dozen times faster than real arithmetic.

Integers are stored in memory as one word (two bytes), corresponding to the 16-bit value of the number. Real numbers, which can be as large as 10 to the power 615, are stored as six bytes: four zero bits, 12 exponent bits and 32 mantissa bits. The exponent is information about the size of the number, and the mantissa stores the value. For example, in base 10, the numbers 2400 and 24 have the same mantissa (24) but a different exponent.

If you are using a very large Real array which runs out of memory, it is worth remembering that an Integer array of the same dimensions will only use one-third of the space.

There are limitations: If you key in and run **listing one**, the program attempts to calculate the hundreth number in the famous Fibonacci sequence: 1, 1, 1, 2, 3,

```
Listing 1 - Fibonacci sequence, SuperBasic, integers.
100 a% = 0
110 b% = 1
120 FOR loop = 1 TO 100
130   c% = a% + b%
140   PRINT 'The '; loop; 'th Fibonacci number is '; c%
150   a% = b%
160   b% = c%
170 END FOR loop
```

```
Listing 2 - Fibonacci sequence, SuperBasic, real numbers.
100 a = 0
110 b = 1
120 FOR loop = 1 TO 100
130   c = a + b
140   PRINT 'The '; loop; 'th Fibonacci number is '; c
150   a = b
160   b = c
170 END FOR loop
```

```
Listing 3: Arithmetic routines & Fibonacci sequence, SuperBasic
100 maxlength = 20
110 zero$ = FILL$ ('0', maxlength)
120 b$ = Real_to_num$ (1)
130 a$ = zero$
140 FOR loop = 1 TO 100
150   c$ = add$ (a$, b$)
160   PRINT 'The '; loop; 'th Fibonacci number is ';
170   writenum c$: REMark removes leading zeroes
180   a$ = b$
190   b$ = c$
200 END FOR loop
210 STOP
220 :
230 DEFine PROCedure writenum (number$)
240 LOCal leading_zero, loop
250 leading_zero = 1
260 FOR loop = 1 to maxlength
270   IF number$ (loop) <> '0' THEN leading_zero = 0
280   IF NOT(leading_zero) THEN PRINT number$(loop);
290 END FOR loop
300 IF leading_zero THEN PRINT '0';
310 PRINT ' ';
320 END DEFine writenum
330 :
340 DEFine FuNction add$ (number1$, number2$)
350 LOCal carry, temp_result, result$(maxlength), loop
360 result$ = zero$
370 carry = 0: REMark no carry in
380 FOR loop = maxlength TO 1 STEP -1
390   temp_result = numer1$(loop) + number2$(loop) + carry
400   IF temp_result > 9 THEN
410     carry = 1
420     temp_result = temp_result - 10
430   ELSE
440     carry = 0
450   END IF
460   result$(loop) = temp_result
470 END FOR loop
480 RETurn result$
490 END DEFine add$
500 :
510 DEFine FuNction shift_left$ (number$)
520 LOCal result$
530 result$ = number$ (2 TO maxlength) & '0'
540 RETurn result$
550 END DEFine shift_left$
560:
570 DEFine FuNction multiply$ (number1$, number2$)
580 LOCal times_table$ (9, maxlength), loop, result$
590 result$ = zero$
600 FOR loop = 0 TO 9
610   times_table$ (loop) = result$
620   result$ = add$ (result$, number1$)
630 END FOR loop
640 FOR loop = 1 TO maxlength
650   result$ = shift_left$ (result$)
```

```
660   result$ = add$( result$,  times_table$( number2$(loop) ) )
670 END FOR loop
680 RETurn result$
690 END DEFine multiply$
700 :
710 DEFine FuNction Real_to_num$ (real)
720 LOCal result$, temp
730 real = ABS (real)
740 result$ = ''
750 REPeat until_real_is_zero
760   temp = INT( real/10)
770   result$ = (real - (10 * temp) ) & result$
780   real = temp
790   IF real < 1 then EXIT until_real_is_zero
800 END REPeat until_real_is_zero
810 result$ = FILL$ ('0', maxlength - LEN(result$) ) & result$
820 RETurn result$
830 END DEFine Real_to_num$
840 :
850 DEFine FuNction negate$ (number$)
860 LOCal loop, result$
870 result$ = zero$
880 FOR loop = 1 TO maxlength
890   result$ (loop) = 9 - number$ (loop)
900 END FOR loop
910 FOR loop = 1 TO maxlength
920   IF result$ (loop) <> '9' THEN
930     result$ (loop) = result$ (loop) + 1
940     EXIT loop
950  END IF
960  result$ (loop) = '0'
970 END FOR loop
980 RETurn result$
990 END DEFine negate$
1000 :
1010 DEFine FuNction subtract$ (number1$, number2$)
1020 RETurn add$ (number1$, negate$ (number2$) )
1030 END DEFine
1040 :
```

5, 8 etc, where each number is the sum of its two predecessors. You will notice that it soon stops with an error — the limit of 32767 has been passed. This shows that although all the Fibonacci numbers are integers, SuperBasic's implementation of Integer arithmetic is not good enough for our purposes.

Now try **listing two**. This time you will find that the program continues to its goal, but after about the fiftieth number, instead of printing all the digits, the output is truncated to six significant digits and an E number. So we have the hundredth Fibonacci number, but only approximately. one of the limitations of Real arithmetic is limited precision: the QL performs calculations only to eight figures, and only shows six of them.

Listing three solves the above problems. Each number is stored as a string; by changing the maximum length of the string, you can change the precision of the arithmetic. It's as simple as that. If you have enough memory, you could work right up to the SuperBasic maximum string length — imagine calculation on 32767 digit numbers!

The function add$ uses the same algorithm we humans use for adding two numbers — starting from the right, with

Listing 4: Arithmetic routines in 68008 Assembly language.

```
OBJECT......LABEL...MEM...OP.......ARGS.........COMMENT.....................
                           ;
                           ; ZEROISE ROUTINE: TAKES NUMBER D1 AND ZEROES IT
                           ;
2801          ZEROISE 0000 MOVE.L   D1,D4         ;
6100 008A             0002 BSR      GETARG        ; GET ADDRESS OF NUMBER IN A4

5385                  0006 SUBQ.L   #1,D5         ; MAXLENGTH - 1

18FC 0000     Z_LOOP  0008 MOVE.B   #0,(A4)+      ; ZEROISE D5 BYTES
51DC FFFA             000C DBRA     D5,Z_LOOP     ; DO IT

7000                  0010 MOVEQ    #0,D0         ; NO ERROR
4E75                  0012 RTS                    ; RETURN TO SUPERBASIC

                           ;
                           ; ADD ROUTINE: ADDS D1 TO D2, RESULT IN D3
                           ;
6100 0088     ADD     0014 BSR      GET3ARGS      ; GET ADDRESSES OF #D1,D2,D3

7C00                  0018 MOVEQ    #0,D6         ; NO CARRY IN

D3C5                  001A ADDA.L   D5,A1         ; POINT TO
D5C5                  001C ADDA.L   D5,A3         ;   END OF
D7C5                  001E ADDA.L   D5,A3         ;     EACH NUMBER

7E00                  0020 MOVEQ    #0,D7         ; D7 = RESULT DIGIT
5385                  0022 SUBQ.L   #1,D5         ; MAXLENGTH - 1

1E21          A_LOOP  0024 MOVE.B   -(A1),D7      ; ADD..
DE22                  0026 ADD.B    -(A2),D7      ;    ..UP..
DF06                  0028 ADD.B    D6,D7         ;        ..THIS DIGIT..

7C00                  002A MOVEQ    #0,D6         ; NO CARRY..
0C07 000A             002C CMPI.B   #10,D7        ;     ..UNLESS RESULT>10
6B06                  0030 BMI.S    NOCARRY       ;

7C01                  0032 MOVEQ    #1,D6         ; SET CARRY FOR NEXT TIME
0407 000A             0034 SUBI.B   #10,D7        ;
```

```
1707          NOCARRY 0038  MOVE.B   D7,-(A3)      ; STORE RESULT

51CD FFE8             003A  DBRA      D5, A_LOOP    ; REPEAT FOR EACH DIGIT

7000                  003E  MOVEQ     #0,D0         ; NO ERROR
4E75                  0040  RTS                     ; RETURN TO BASIC

                            ;
                            ; SHIFTLEFT NUMBER D1
                            ;
3801          SH_LEFT 0042  MOVE.W   D1,D4         ; GET POINTER TO NUMBER
6101 0048             0044  BSR      GETARG        ; IN A4

5585                  0048  SUBQ.L   #2,D5         ; MAXLENGTH -2
264C                  004A  MOVE.L   A4,A3         ; A4=BEGINNING OF NUMBER
528B                  004C  ADDQ.L   #1,A3         ; A3= A4+1

18DB          SH_LOOP 004E  MOVE.B   (A3)+,(A4)+  ; COPY IT
51CD FFFC             0050  DBRA     D5,LOOP3      ;

18FC 0000             0054  MOVE.B   #0,(A4)+      ; INSERT A TRAILING ZERO

7000                  0058  MOVEQ    #0,D0         ; NO ERROR
4E75                  005A  RTS                    ; RETURN TO BASIC

                            ;
                            ; WRITENUM #CHANNEL D1, NUMBER D2
                            ;
C2FC 0028     WRITNUM 005C  MULU.W   #40,D1        ; (CH_LENCH ) GET POINTER..
D2AE 0030             0060  ADD.L    48(A6),D1     ; (BV_CHBAS ) ..TO BASIC CHAN..
B2AE 0034             0064  CMP.L    52(A6),D1     ; (BV_CHP   ) ..#D1 IN A0.
6220                  0068  BHI.S    ERR_NO        ;
2076 1800             006A  MOVE.L   0(A6,D1.L),A0;

2802                  006E  MOVE.L   D2,D4         ;
6100 001C             0070  BSR      GETARG        ; GET POINTER IN A4

5385                  0074  SUBQ.L   #1,D5         ; MAXLENGTH -1
76FF                  0076  MOVEQ    #-1,D3        ; INFINITE TIMEOUT

7200          W_LOOP  0078  MOVEQ    #0,D1         ;
121C                  007A  MOVE.B   (A4)+,D1      ; BYTE TO SEND
0601 0030             007C  ADDI.B   #48,D1        ; CONVERT DIGIT TO ASCII
7005                  0080  MOVEQ    #5,D0         ; IO_SBYTE
4E43                  0082  TRAP     #3            ; QDOS TRAP TO PRINT CHAR
51CD FFF2             0084  DBRA     D5,W_LOOP     ; REPEAT FOR EACH DIGIT

4E75                  0088  RTS                    ; RETURN TO BASIC

70FA          ERR_NO  008A  MOVEQ    #-6,D0        ; ERROR - CHANNEL NOT OPEN
4E75                  008C  RTS                    ; RETURN WITH ERROR

                            ;
                            ; GETARG - TAKES NUMBER D4, RETURNS IN A4 THE
                            ; ADDRESS OF THE CORRESPONDING 'LONG NUMBER'
49FA 0046     GETARG  008E  LEA.L    NUMBASE,A4    ;
7A00                  0092  MOVEQ    #0,D5         ;
3A2C FFFE             0094  MOVE.W   -2(A4),D5     ; MAXLENGTH

C8C5                  0098  MULU.W   D5,D4         ; MAXLENGTH * NUMBER
D9C4                  009A  ADDA.L   D4,A4         ; .. + NUMBASE

4E75                  009C  RTS                    ;

                            ;
                            ;GET3ARG: TAKES 3 ARGS D1,D2,D3, RETURNS ADDRESSES
                            ;OF CORRESPONDING LONG NUMBERS IN A1,A2,A3
2801          GET3ARG 009E  MOVE.L   D1,D4         ;  GETARG TAKES D4 &..
6100 FFEC             00A0  BSR      GETARG        ;  .. RETURNS A4, BEING..
224C                  00A4  MOVE.L   A4,A1         ;  .. THE POINTER TO THE..

2802                  00A6  MOVE.L   D2,D4         ;  .. BASE OF NUMBER#D4.
6100 FFE4             00A8  BSR      GETARG        ;
244C                  00AC  MOVE.L   A4,A2         ;
```

```
2803                    00AE  MOVE.L   D3,D4          ;
6100 FFDC               00B0  BSR      GETARG         ;
264C                    00B4  MOVE.L   A4,A3          ;

4E75                    00B6  RTS                     ;

                              ;
                              ; COPYNUM D3 TO NUM D4
                              ;
2E04          COPYNUM   00B8  MOVE.L   D4,D7          ;

6200 FFD2               00BA  BSR      GETARG         ; ARG D4
204C                    00BE  MOVE.L   A4,A0          ; STORE POINTER

2803                    00C0  MOVE.L   D3,D4          ; ARG D4
6100 FFC                00C2  BSR      GETARG         ;

5385                    00C6  SUBQ.L   #1,D5          ; MAXLENGTH -1

18D8          C_LOOP    00C8  MOVE.B   (A0)+,(A4)+    ; COPY #D4 TO #D3
51CD FFFC               00CA  DBRA     D5,C_LOOP      ; REPEAT FOR REST OF NUMBER

2807                    00CE  MOVE.L   D7,D4          ;

7000                    00D0  MOVEQ    #0,D0          ; NO ERROR
4E75                    00D2  RTS                     ; RETURN TO BASIC

0014          MAXLEN    00D4  DC.W     20             ;
              NUMBASE   00D6
                              END
```

Listing 5: SuperBasic + Machine Code, Fibonacci sequence using Large Numbers.

```
100 Set_up_Mcode
110 CLS
120 Real_to_num 1, 1
130 zeroify 2
140 FOR loop = 1 to 100
150   add 1, 2, 3
160   PRINT 'The '; loop; 'th Fibonacci number is exactly ';
170   writenum #1, 3
180   copynum 1, 2
190   copynum 2, 3
200 END FOR loop
210 STOP
220 :
1000 REMark Large Integer Arithmetic Routines
1010 DEFine PROCedure zeroify (number)
1020 CALL BASE, number
1030 END DEFine zeroify
1040 :
1050 DEFine PROCedure writenum (channel, number)
1060 CALL BASE + 92, channel, number: PRINT #channel
1070 END DEFine writenum
1080 :
1090 DEFine PROCedure copynum (num1, num2)
1100 CALL BASE + 184, 0, 0, num1, num2
1110 END DEFine copynum
1120 :
1130 DEFine PROCedure add (num1, num2, num3)
1140 CALL BASE + 20, num1, num2, num3
1150 END DEFine add
1160 :
1170 DEFine PROCedure shift_left (number)
1180 CALL BASE + 66, number
1190 END DEFine shift_left
1200 :
1210 DEFine PROCedure multiply (num1, num2, num3)
1220 LOCal loop, num2base
1230 zeroify 10
1240 FOR loop = 11 TO 19
1250   add num1, loop-1, loop
1260 END FOR loop
1270 zeroify num3
1280 num2base = NUM_BASE + maxlength * num2
1290 FOR loop = 1 TO maxlength
1300   shift_left num3
1310   add 10 + peek (loop + num2base - 1), num3, num
1320 END FOR loop
1330 END DEFine multiply
1340 :
1350 DEFine PROCedure Real_to_num (real, number)
1360 LOCal PTR, loop: PTR = NUM_BASE + number * maxlength
1370 real = abs (real)
1380 zeroify number
1390 FOR loop = 1 to LEN(real)
1400   temp = INT (real/10) : REMark don't use DIV
1410   POKE PTR + maxlength - loop , real - (10 * temp)
1420   real = temp
1430 END FOR loop
1440 END DEFine Real_to_num
```

carries, etc. The function multiply$ uses a modified version of the 'long multiplicaiton' algorithm. This is not the most efficient algorithm but it is easy to understand. First it creates a "time-table" of the first argument. Then it goes along the second argument from left to right, adding the correct multiple of the first argument (found by looking up in the table) to the result, then multiplying the result by ten (by shifting it left). Try following this yourself, by hand, if you don't understand it at first.

The subtract$ function works by negating the second number, then adding this to the first. To negate a base-ten number, you substract each digit from nine, then add one to the result. This technique has been used for years to store negative binary numbers, but is rarely used for base ten.

Seconds

If you run listing three, you will finally get your result: the hundredth Fibonacci number is calculated in seconds. These routines are fine if all you want to do is add numbers up, but if you try some multiplication you may have a long wait. The multiplication algorithm above perfoms ten additions to set up the table, then as many additions as there are digits in the result. Thus the performance of the multiplication routine depends critically upon the addition routine. Because of this I have recorded the addition routine in machine-code.

Accessing SuperBasic strings from machine code would rob the program of the speed gains it is trying to achieve,so the second version of the program stores

the numbers directly in memory. The algorithms used are the same as the ones used in the SuperBasic implementation, so I shall not describe them agin. **Listing four** contains the source code, well documented. **Listing five** loads the machine code routines into memory.

Calculator

To show off the new routines, **listing six** turns the QL into a pocket calculator. The variable maxlength must not be changed in the middle of the program, and remember that because of the ten numbers used by the multiply routine, you must have space in memory for at least twenty numbers. So, on an unexpanded QL, you are limited to operations on numbers only about 4000 digits long. This should be sufficient for most calculations.

If the program had been left entirely in SuperBasic, it would have worked correctly, but not fast enough (especially for multiplications) for any practical use. If one had attempted to recode the entire program in assembly languauge, the program would probably never have been finished, as it takes a long time to write and debug machine code. However, by coding first in a high-level language (SuperBasic), the replacing the critical parts of fast machine code, we get the best of both worlds — the most power from the QL, with the least effort.

```
1450 :
1460 DEFine PROCedure subtract (num1, num2, num3)
1470 LOCal loop, num2base
1480 num3base = NUM_BASE + maxlength * num3
1490 copynum num3, num2
1500 FOR loop = 0 TO maxlength - 1
1510   POKE (num2base + loop) , 9 - PEEK ( num3base + loop)
1520 END FOR loop
1530 Real_to_num 1,10
1540 ADD 10, num3, 11
1550 ADD num1, 11, num3
1560 END DEFine subtract
1570 :
1580 DEFine PROCedure Set_up_Mcode
1590 LOCal loop, checksum, total, loop2, data_item
1600 REMark This procedure must always be run FIRST in any program.
1610 maxlength = 20 : REMark can change this, min = 2, max = thousands
1620 BASE = RESPR ( 216 + 20 * maxlength )
1630 RESTORE
1640 FOR loop = BASE TO (BASE + 204) STEP 12
1650   total =0
1660   FOR loop2 = loop TO (loop + 10) STEP 2
1670     POKE_W loop2, data_item
1680     total = total + data_item
1690   END FOR loop2
1700   READ checksum
1710   IF checksum <> total THEN PRINT 'DATA ERROR':STOP
1720 END FOR loop
1730 NUM_BASE = BASE + 214
1740 POKE_W NUM_BASE - 2 , maxlength
1750 RETurn
1760 DATA 10241, 2432, 138, 21381, 6396, 0, 62988
1770 DATA 20941, -6, 28672, 20085, 24832, 136, 94660
1780 DATA 31744, -11323, -10811, -10299, 32256, 21381, 52948
1790 DATA 7713, -8670, -8442, 31744, 3079, 10, 25434
1800 DATA 27398, 31745, 1031, 10, 5895, 20941, 87020
1810 DATA -24, 28672, 20085, 14337, 24832, 72, 87934
1820 DATA 21893, 9804, 21131, 6363, 20941, -4, 80128
1830 DATA 6396, 0, 28672, 20085, -15620, 40, 39573
1840 DATA -11602, 48, -19794, 52, 25120, 8310, 2134
1850 DATA 6144, 10242, 24832, 28, 21381, 30643, 93090
1860 DATA 29184, 4636, 1537, 48, 28677, 20035, 84117
1870 DATA 20941, -14, 20085, 28922, 20085, 18938, 108957
1880 DATA 70, 31232, 14892, -2, -14139, -9788, 22265
1890 DATA 20085, 10241, 24832, -20, 8780, 10242, 74160
1900 DATA 24832, -28, 9292, 10243, 24832, -36, 69135
1910 DATA 9804, 20085, 11780, 24832, -46, 8268, 74723
1920 DATA 10243, 24832, -54, 21381, 6360, 20941, 83703
1930 DATA -4, 10247, 28672, 20085, 20, 0, 59020
1940 END DEFine
```

```
Listing 6: Cal-QL-ator; Needs listing 5 to run.
100 Set_up_Mcode
110 Set_up_Screen
120 REPeat main_loop
130   key = CODE (INKEY$ (-1) )
140   SELect ON key
150     = 48 TO 57: number_pressed
160     = equals, enter : equals_pressed
170     = plus, minus, times : operator_pressed
180   END SELect
190 END REPeat main_loop
200 :
210 DEFine PROCedure number_pressed
220 IF op = equals THEN
230   op = plus
240   zeroify displayed_num
250   zeroify current_num
260 END IF
270 shift_left current_num
280 POKE NUM_BASE + maxlength * (current_num + 1) - 1, key - 48
290 PRINT CHR$ (key);
300 END DEFine
310 :
320 DEFine PROCedure work_it_out
330 SELect ON op
340   = plus   : add      displayed_num, current_num, result
350   = minus  : subtract displayed_num, current_num, result
360   = times  : multiply displayed_num, current_num, result
370   = equals : copynum result, current_num
380 END SELect
390 END DEFine
400 :
```

```
410 DEFine PROCedure operator_pressed
420 PRINT ' '; CHR$(key); ' ';
430 work_it_out
440 copynum displayed_num, result
450 ZEROIFY current_num
460 op = key
470 END DEFine
480 :
490 DEFine PROCedure equals_pressed
500 PRINT ' = ';
510 work_it_out
520 writenum #1, result
530 copynum current_num, result
540 op = equals
550 END DEFine
560 :
570 DEFine PROCedure Set_up_screen
580 LOCal loop, loop2: MODE 4
590 equals = CODE ('='): plus = CODE ('+'):  enter = 10
600 minus  = CODE ('-'): times = CODE ('*')
610 result = 0: displayed_num = 1: current_num = 2: op = plus
620 zeroify result: zeroify displayed_num: zeroify current_num
630 OPEN #1, con_400x240a56x16: PAPER 4: CLS
640 CSIZE 3,1: INK 0
650 AT 6,0: PRINT '  7  8  9  -  *' \\ '  4  5  6  +' \\ '  1  2  3  0  ='
660 REMark 2 spaces between each symbol in previous line
670 FOR loop = 110 TO 230 STEP 40: BLOCK 236, 4, 16, loop, 0
680 FOR loop = 10 TO 294 STEP 48: BLOCK 6, 124, loop, 110, 0
690 CSIZE 0,0: AT 10,2: PRINT 'Cal-QL-ate: Max length = '; maxlength
700 WINDOW 390, 94, 60, 18: INK 7: PAPER 0: CLS: BORDER 2,2
710 END DEFine
720 :
```




IF AN ADVERT IS IN PRINT, IS IT PROPER?

Most advertisements are perfectly proper. A few are not.

The Advertising Standards Authority not only monitors over 850 advertisements every month, it ensures compliance with the rules in the strict Code of Advertising Practice.

So when you question an advertiser, they have to answer to us. To find out more about the role of the ASA, please write to Advertising Standards Authority, Department X, Brook House, Torrington Place, London WC1E 7HN.

ASA

This space is donated in the interests of high standards in advertisements.

SOFTWARE FILE

SPELLBOUND-2

INFORMATION:
Program: ***Spellbound 2***
Price: £50
Supplier: Sector Software, Unit 13, Centurion Way Industrial Estate, Farington, Leyland, Lancs PR5 2GU.
Tel: 0772 454328

The original version of *SpellBound* obviously met a considerable need within the QL community. There were many interested spectators around the Sector Software stand at the Microfair when it was introduced, and the program has proved to be a good seller. Surprisingly, it has sold much better than *FlashBack*, which seems to fill a more important niche in the software market. Maybe this goes to show how little software writers, suppliers, and reviewers, know of what users really want.

One feature of *SpellBound 1* which gave it considerable appeal was the immediate nature of the spelling checking; it was right there with the writer, following every keystroke and commenting as soon as it didn't like one of them. This was a feature I had difficulty getting along with, and no doubt many other users also made their feelings known; we wanted to be able to check existing files, rather than new ones. There was little doubt retrospective checking would be a feature of a revised SpellBound, the only question being when, and if, the revision would appear. Since the original introduction, QJump had looked at SpellBound, found various aspects of it that were considered unsatisfactory, and introduced *QTyp*.

Among the advantages QTyp has is the ability to check existing files, at high speed. Although *FileBound* came along and enabled SpellBound to handle existing files, the process can hardly be called speedy, and it has a few quirks. Fortunately for Sector, SpellBound had established a sizeable base of faithful users, and QJump never seemed to make any serious attempt to market QTyp to the total QL market; this gave plenty of time to come up with version 2 of SpellBound.

There has been one other competitor, though. This is still, in essence, QTyp, but integrated into the *text*^{87} word-processing program. This version of QTyp is somewhat simpler than the stand-alone version; it works well, but it does not provide as-you-type checking of spelling and is really only intended for use with *text*^{87} _T87 files. You can actually check anything that can be loaded into *text*^{87}, and that includes *Quill* _DOC files, but this really doesn't basically affect the market for SpellBound 2, which must largely be those users who have never changed from Quill. You can use the standard 2.35 Quill, or the version patched to run from EXEC as well as EXEC_W. Quill treated with Turbo-Plus is also acceptable.

There should not be any compatibility problems with earlier versions of Quill; certainly 2.30 should be alright, but it might be worth asking about versions earlier than that (better still, get rid of them!) Although it is aimed at Quill users, the program can be used with other programs; it works in both concurrent and retrospective modes with *text*^{87} and *The Editor*, but the user needs to bear in mind that there are codes within a _T87 file which are not displayed on-screen, but will be seen by SpellBound. There does not appear to be any problem using the program in a system fitted with a *Minerva* rom, although revisions of the latter come out faster than software writers can be expected to keep pace with.

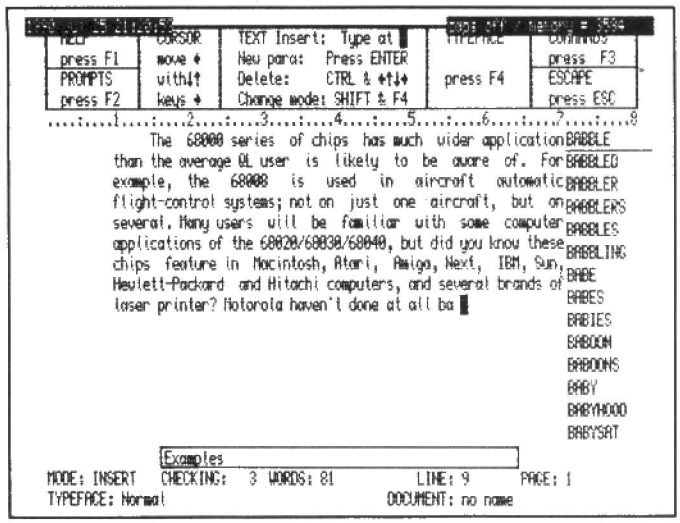
What is new or improved for the user? Retrospective checking has been mentioned, and that has to be the major new feature. During concurrent checking, the tendency the program had to switch off when it detected non-alphanumeric keypresses, and the need to use the command keying to restart it, was a constant irritation. This situation has now been improved, with there being less tendency to drop out, and restarting occurring automatically when Space is pressed (and most users won't type far without hitting Space).

As befits this bigger-is-better era, an alternative, larger dictionary is provided, with 50,000-plus words as against the 30,000 of the standard one. You can use whichever one you prefer; put another way, whichever your QL's memory can hold. "Sinclair" is in the standard dictionary, suggesting that it is of more practical use than most. Loading of the dictionary file is quicker, and automatic; the file "dictionary" is loaded by default, but you can choose to rename any other dictionary file to that name if you want to.

Another big improvement shows when saving the current dictionary — the act of saving no longer disables spell-checking. There is no need to read the dictionary back in again to do some more checking, as it remains there after a save; this means that you can make precautionary saves whenever you wish, without the action disrupting checking work.

The checking modes have been reduced by one, Mode 5 being removed because it was considered of little interest to users. The modes are selected by CTRL F1 to F4, with the default being Mode 3 (CTRL F3). SpellBound now has three states: off, with nothing happening or visible; on, but "asleep", with its cursor flashing; on, and checking. You get from the sleep to active state by pressing Space, and back to the sleep state by ALT-ESC. The box for displaying suggested spellings can be moved around the screen, and re-sized, using CTRL-M followed by the standard cursor-and-ALT key procedure (as in FlashBack).

Preparation of a working copy is straightfoward. You can incorporate SpellBound into a system boot routine, to have it available all the time. The program file is about 23 KB in size, and the two dictionary files are 59 KB and 107 KB. When loaded, the space taken is about 170 KB with the standard dictionary, and 250 KB with the larger one. The main reason the figures don't add up is that the files have to be uncompressed in memory for use, and they then take up roughly twice as much space



Bryan Davies tries out a new attack on the typos.

as they do on disk. Your QL definitely needs memory expansion!

SpellBound is a stand-alone program, so you don't have to run your WP program to check spelling in existing documents. You simply load SpellBound, and start it by keying CTRL O, as before. Keying CTRL F then brings up a menu of options for checking existing files — MARK, Y/N ADD, AUTO ADD, UNMATCHED, START HERE, ESC. You key the first letter(s) of the required option(s), then press ENTER, to initiate action. ESC gets you back out of the menu. Any tendency to forget to key ENTER is counteracted by that word moving from side to side on the menu, then growing in size and being accompanied by beeping, to attract your attention.

You can select any compatible combination of choices from the menu — for example, MARK and Y/N ADD. If the Y/N ADD option is chosen, every instance of a word unknown to the dictionary produces both that word and a prompt message in the familiar box at the middle, bottom of the screen.

The ("hat") character is displayed at the point in the word where SpellBound decided it was unknown. Being faced with a word and a question concerning it, without seeing the context, makes me uncertain as to how to react; there are differing uses for some combinations of characters, which might lead you to accept/reject them, then find your choice was wrong.

When the document being checked is a Quill one, this difficulty can be overcome by loading the document into Quill beforehand, so that you see the full screen of text as well as the SpellBound window. The text is made to page up the screen to keep in synchronism with what is being checked, by putting Quill into Search mode, a very neat touch. Words are displayed in the actual case and, as before, words beginning with capital letters or ending with full stops bring up additional questions — should the word always start with a capital (proper noun), or end with a full stop (in which case it is an abbreviation)? This function seems very basic to spelling checking, yet only SpellBound (of the checkers I recollect seeing) has it.

When you are not certain of the spelling of a word, the Examples screen can be displayed by keying CTRL E; you can page down the examples by keying SHIFT DOWN, or move one at a time with the up/down cursor keys. If you wish to replace your spelling by that of an example, you move the cursor until the chosen example is highlighted by a white underline, then press ENTER, and that word takes the place of the error word in the text.

The speed of checking a document is basically up to the user — the faster you give answers to the questions, the sooner the job is finished. As the text files are not in memory, the media influences the speed. With hard disk, the delay in moving from one unknown word to the next is slight, but microdrives give noticeable pauses as the next block of text is moved into memory. When there are many unknown words, the overall rate may be only 20-30 words per minute, but it rises to several thousand words per minute if there are few unknown words. Choosing the MARK option demonstrates the speed very effectively. An 1800-word file was marked (which included writing-out the marked version) in 10 seconds — over 10,000 words per minute. This is one of those functions where benchmark speeds are pointless, once a certain performance level has been reached; SpellBound 2 is as fast as it reasonably needs to be.

Documents checked with the MARK option need to be loaded back into Quill, to make any changes deemed necessary. SpellBound helpfully asks if you want to load the document into Quill, immediately the checking is finished, and loading is done for you if you answer "Y" (provided Quill is already running). The marked document has the extension _CHK, but the Quill Load function is quite happy to accept that. Quill is placed automatically in Search and Replace mode, so that it will remove the ("hat") characters. Whenever you need to edit a marked word, you press ESC, make the change, then press F5 to resume the Search and Replace operation (this procedure works only if SpellBound is in Mode 2 or 3). A vetted document can then be re-checked using the AUTO ADD option, which causes all unknown words to be be entered into the dictionary. The speed here is roughly the same as for the MARK option. Overall, you will probably get the job done quicker by keeping it to one operation — Y/N ADD — but the MARK/AUTO ADD route does allow you to see the full text with the marked words, before risking adding anything to the dictionary.

The program is nothing if not flexible. In case the two approaches already mentioned are not to the user's taste, there is the UNMATCHED option. This creates a file containing only those words which were not found in the dictionary. The creation speed is about the same as noted with the other functions. The file can then be loaded into your WP program for editing, and the words added subsequently to the dictionary using the AUTO ADD option. One more option is available. There may be times when you do not want to check the whole of a document, and you can restrict SpellBound's interest to a given section by typing-in "StartHere" at the relevant start-point in the document, and selecting the START HERE option, such as Y/N ADD. The "StartHere" addition is removed from the document automatically during checking.

There seems little reason to comment much on SpellBound 2 as a concurrent checker, as it is basically the same as version 1. There was not much need to make changes, with the obvious exception that the "hypersensitive" behaviour needed calming down somewhat, and that has been achieved. Even if you forget to hit Space to switch checking back on as soon as you have moved the cursor away from the current text-insertion point, it is virtually certain you will hit Space after typing a few more characters so that only an odd word will not be checked. To make certain you don't get into the habit of ignoring the chirps from the QL when an unknown character combination occurs with Mode 3 in use, you can still change to Mode 4 and have the keyboard disabled with the cursor on the questionable character, so that you are forced to pay attention to it. A fast typist is unlikely to accept the intrusion of concurrent checking, but most of us are pretty slow on the keys and will be warned of potential errors before we have typed more than a few other characters. Either way, there is a Mode to suit your style.

A few odd quirks still exist. When trying to load a dictionary file, if the file named does not exist you may have difficulty proceeding; ESC or ENTER fail to get you away from the prompt, and what is required is the up/down cursor key. Hopefully, this point will be made in the instructions (not available during the review). You need to de-activate SpellBound with ALT-ESC before switching to another program, to avoid losing it altogether. It would help if an option were added to ignore specific strings, that occur repeatedly but are not wanted in the dictionary.

SpellBound 2 is a considerable improvement over version 1 and should now be suitable for virtually all Quill users. It is also suitable for The Editor and *text*^{87}, although you don't get some of the nice touches with them that are provided when using Quill. With any spelling checker, the decision as to whether it is needed at all is a personal matter; if you feel sensitive about the impression you give to other people when your letters, etc. contain spelling mistakes — or typographical errors — this program is well worth having. If you don't believe you make "typos", try taking another look at some old documents; if you don't find any mistakes, the chances are you are not looking closely enough. Maybe you would just like something to make Quill a bit more interesting — the cost of SpellBound is still low enough for some users' "discretionary" budgets, although it is quite a bit higher than it was (QL users will have to get used to higher prices, or it won't be worth writers and suppliers producing new things for the smaller market).

BRAND NEW
MICRODRIVE CARTRIDGES
£2.00 EACH ANY QUANTITY

Sector Software

The best programs and peripherals for the QL

QZ/QL to Z88 File Transfer
Software and cable to connect the Z88 and QL and transfer any files between them. Includes Archive to Pipedream and back conversion routines. **£25**

Amiga to Z88 File Transfer
Software and cable to connect the Z88 and Amiga and transfer any files between them **£25**

Spellbound
A spelling checker that checks your spelling AS YOU TYPE. Based on a 30,000 word dictionary, works with Quill or The Editor V1.17 onwards on the expanded QL. **£30**

Taskmaster
A brilliant multitasking front end system which lets you use the QL as a serious machine. Multitask many programs at once. **£25**

Files 2
File handling utility with scores of features. Written by Peter Jeffries. Ideal enhancement for Taskmaster users. **£12**

Write Turn
Turn spreadsheets and documents on their sides with this excellent utility, works on Epson and compatible printers. **£12**

QL World Index
A complete index to the contents of QL World from its start to May 1988. Find articles and reviews in seconds, 160K+ of data compressed to fit into a 128K QL. **£6**

Flashback
A very fast and slick database which has very few limitations. Will also convert Archive files. **£25**
Flashback Special Edition is a greatly advanced version with lots of extra features including report generator, mail merge, label printing, etc. **£40**

Touch Typist
Excellent typing tutor that works. 200 lessons, graph of your progress, adjustable difficulty levels. **£12**

Ferret
Find lost files fast with this file search utility which will read all your files on disk or mdv looking for a match with your search text. **£12**

STD Index
This index to all the dialling codes in the country executies from disk in 15 seconds. Know the place and it will tell you the number, know the number and it will tell you the place!
(Expanded QL only.) **£12**

Page Designer 2
This is a full feature desktop publisher that has to be seen to be believed. Ask for full details of this system and its support programs. **£35**

Phillips CM8833 Colour Stereo Monitor
A stereo monitor for the QL, Amiga, ST or almost any computer. **£260**

Cambridge Satellite TV System 2
48 channel stereo infra red remote control receiver with 60cm square dish. Receive all the current and planned future Astra transmissions. Now available for immediate delivery. Contact us for a leaflet on this system or call in and see the quality for yourselves.

SYSTEM II 1.8 DB LNB £299
GOLD SYSTEM 1.5 DB LNB £349

DRAMATIC PRICE REDUCTIONS ON STAR PRINTERS!

Star LC10 printer	£190
Star LC10 colour printer	£230
Star LC2410 (24 pin) printer	£274
(Please add £6 if carrier delivery required)	
Z88 Computer	£230
Z88 All in one pack (Z88 128K RAM, carry case, power supply, batteries, user manual)	£287
QL Keyboard membrane	£7
QL Service Manual	£25
ZX8301 ULA	£10
3.5in. DSDD disks (each)	£1.25
Cartridge Doctor (rescue corrupt MDV's)	£16

New Price List available – Send for your free copy

Access VISA

Sector Software

MasterCard EUROCARD

Unit 13, Centurion Way Industrial Estate, Farington, Leyland, Lancs. PR5 2GU
Tel: (0772) 454328/452414 (2 lines), Fax: (0772) 454680